Algorismes d'integració eficients per la simulació computacional de sistemes físics atòmics i moleculars

Autor: Bernat Díaz Ferran
Director: Jordi Marti
Tutor a GEP: Fernando Barrabes Nadal
Especialitat: Computació
18 de setembre de 2019

Índex

Context

Aquest projecte es fa a la universitat en el departament de física i és de l'especialitat de computació.
S'enfoca en el problema de càlcul numèric. El càlcul numèric es una branca de les matemàtiques encarregada de dissenyar algorismes per simular aproximacions a problemes matemàtics. Dintre del càlcul numeric trobem varies àrees d'estudi entre elles el càlcul d'equacions diferencials que és el que ens pertoca.
En el càlcul d'equacions diferencials coneixem el punt inicial i una equació que descriu la pendent de la funció que volem aproximar en un punt. La funció que volem aproximar és doncs desconeguda. El mètode de càlcul consisteix en a partir del valor inicial (x0) trobar el següent valor passat un temps molt petit (dt), trobant així el valor x1 i repetir aquest procés utilitzant com a valor inicial el nou valor trobat (x1) i calcular amb aquest el següent valor (x2). És a dir a partir d'un valor xi aplicant la funció amb un instant de temps petit (dt) podem trobar el següent valor xi+1.
L'error doncs bé donat per la magnitud de l'instant de temps.
En alguns sistemes senzills, com per exemple el moviment sota el camp gravitatori, el valor real sense error el coneixem ja que disposem de les equacions del moviment de Newton:
$r(t) = r0 + v0\ t + ½\ a\ t^2$
$v(t) = v0 + a\ t$
L'objectiu del càlcul numèric és doncs aproximar aquestes equacions.
S'utilitzarà l'algorisme de Verlet (Verlet, 1967), aquest es basa en aproximar les equacions del moviment mitjançant les series de Taylor, aquestes aproximen una funció continua mitjançant la següent series de potències:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots$$

Així al aplicar les series de Taylor a les equacions de moviment queda:
$r(t + dt) = 2r(t) - r(t - dt) + dt^2a(t)$
$v(t) = (r(t + dt) - r(t - dt)) / 2dt$
Aquest algorisme però te imprecisions numèriques ja que en l'equació de la posició (r(t + dt)) el terme $dt^2a(t)$ és molt petit comparat amb els altres 2.
S'han fet modificacions d'aquest algorisme i ens centrarem en una d'aquestes anomenat "half step 'leap-frog' scheme" (Hockney 1970; Potter 1972):
$r(t + dt) = r(t) + dt\ v(t + ½\ dt)$
$v(t + ½\ dt) = v(t - ½\ dt) + dt\ a(t)$
$v(t) = ½(v(t + ½\ dt) + v(t - ½\ dt))$
El problema a resoldre es doncs aplicar aquest algorisme a varis sistemes físics i finalment modificar aquest algorisme per millorar-ne l'eficiència o precisió.

No s'ha aconseguit millorar l'eficiència modificant l'algorisme de Verlet raó per la qual s'ha partit d'un altre esquema, els mètodes lineals de múltiples pasos.
En el càlcul d'equacions diferencials coneixem el valor inicial d'una funció i la seva derivada:

$$y' = f(t, y), \quad y(t_0) = y_0.$$

Els mètodes lineals de múltiples pasos utilitzen els valors de s pasos previs de y i f(t,y) per trobar el següent valor de y.

$$\begin{aligned} & y_{n+s} + a_{s-1} \cdot y_{n+s-1} + a_{s-2} \cdot y_{n+s-2} + \cdots + a_0 \cdot y_n \\ & \quad = h \cdot (b_s \cdot f(t_{n+s}, y_{n+s}) + b_{s-1} \cdot f(t_{n+s-1}, y_{n+s-1}) + \cdots + b_0 \cdot f(t_n, y_n)) \\ & \Leftrightarrow \sum_{j=0}^{s} a_j y_{n+j} = h \sum_{j=0}^{s} b_j f(t_{n+j}, y_{n+j}), \end{aligned}$$

Dintre dels mètodes lineals de múltiples pasos hi ha 3 families: Adams-Bashforth, Adams-Moulton i BDF.
En els mètodes de Adams-Bashforth els coeficients aj son

$$a_{s-1} = -1 \; a_{s-2} = \cdots = a_0 = 0$$

El coeficient bs és 0.
Els altres coeficients bj es determinen mitjançant interpolació polinomica per trobar el polinomi p de grau s - 1 tal que:

$$p(t_{n+i}) = f(t_{n+i}, y_{n+i}), \qquad \text{for } i = 0, \ldots, s-1.$$

La solució mitjançant la fórmula de Lagrange és:

$$p(t) = \sum_{j=0}^{s-1} \frac{(-1)^{s-j-1} f(t_{n+j}, y_{n+j})}{j!(s-j-1)!h^{s-1}} \prod_{\substack{i=0 \\ i \neq j}}^{s-1} (t - t_{n+i}).$$

El polinomi p es una bona aproximació local de la funció f en y' = f (t, y) així que podem utilitzar l'equació:

$$y_{n+s} = y_{n+s-1} + \int_{t_{n+s-1}}^{t_{n+s}} p(t)\, dt.$$

Al substituir el polinomi queda el metode d'Adams-Bashforth:

$$b_{s-j-1} = \frac{(-1)^j}{j!(s-j-1)!} \int_0^1 \prod_{\substack{i=0 \\ i \neq j}}^{s-1} (u+i)\, du, \qquad \text{for } j = 0, \ldots, s-1.$$

Al substituir f (t, y) per p(t) estem cometent un error d'ordre h^s.
Els mètodes d'Adams-Moulton són similars als mètodes d'Adams-Bashforth però sense la restricció que bs = 0.
El polinomi interpolador utilitza el punt t_n a part dels punts t_n - 1 .. t_n-s.

$$b_{s-j} = \frac{(-1)^j}{j!(s-j)!} \int_0^1 \prod_{\substack{i=0 \\ i \neq j}}^{s} (u+i-1)\, du, \qquad \text{for } j = 0, \ldots, s.$$

Els BDF “Backward differentiation formula” són mètodes implícits amb

$$b_{s-1} = \cdots = b_0 = 0$$

S’utilitzen per resoldre “stiff differential equations” que són equacions per les que la seva resolució amb certs mètodes és numéricamente inestable.

Justificació

S'ha intentat utilitzar l'algorisme de Verlet i modificar-lo per millorar l'eficiència o precisió. No s'ha aconseguit motiu pel que s'intentarà crear un altre algorisme amb precisió i eficiencia semblant utilitzat un mètode lineal de múltiples pasos.

L'algorisme més simple per resoldre equacions diferencials és l'algorisme d'Euler. Aquest serveix per resoldre equacions diferencials ordinàries donat un valor inicial. L'algorisme d'Euler consisteix en dividir l'interval de l'eix d'abscisses que es vol aproximar en n parts. Després començant pel valor (x0, y0) es procedeix de la següent forma:
On h és l'amplada dels intervals de l'eix d'abscisses és a dir xi+1 - xi

$$y_1 = y_0 + hf(x_0, y_0)$$
$$y_2 = y_1 + hf(x_1, y_1)$$
$$\vdots$$
$$y_{i+1} = y_i + hf(x_i, y_i)$$
$$\vdots$$
$$y_n = y_{n-1} + hf(x_{n-1}, y_{n-1})$$

https://es.wikipedia.org/wiki/M%C3%A9todo_de_Euler
El mètode d'Euler pertany als mètodes de Runge-Kutta que són un conjunt de mètodes genèrics iteratius de resolució numèrica d'equacions diferencials.
El terme d'ordre s es calcula:

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

On h torna a ser l'amplada dels intervals de l'eix d'abscisses.
Els coeficients ki són termes d'aproximació intermitjos avaluats en f de forma local.

$$k_i = f\left(t_n + h\,c_i\,, y_n + h \sum_{j=1}^{s} a_{ij} k_j\right) \qquad i = 1, \ldots, s.$$

aij, bi i ci són coeficients propis de l'esquema numèric escollit depenent de la regla de quadratura utilitzada.
L'algorisme de Verlet "half step 'leap-frog' scheme" (el que farem utilitzarem) fa servir les posicions, velocitats i acceleracions en diferents instants de temps, s'ha proposat l'algorisme que resol aquest problema (Swope, Andersen, Berens, Wilson 1982):
$r(t + dt) = r(t) + dt\, v(t) + ½\, dt^2\, a(t)$
$v(t + dt) = v(t) + ½\, dt(a(t) + a(t + dt))$
L'algorisme funciona de la següent forma:

- Es calcula la nova posició amb la fórmula:
  $r(t + dt) = r(t) + dt\, v(t) + ½\, dt^2\, a(t)$
- Es calcula la velocitat a mig pas amb la fórmula:
  $v(t + ½\, dt) = v(t) + ½\, dt\, a(t)$
- Es calculen les forces i acceleracions en el temps $t + dt$
- Es calcula la velocitat en l'instant $t + dt$ amb la fórmula:
  $v(t + dt) = v(t + ½\, dt) + ½\, dt\, a(t + dt)$

Beeman (1976) va investigar varis algorismes entre ells el següent (Sangster i Dixon 1976; Hockney i Eastwood 1981):
$r(t + dt) = r(t) + dt\, v(t) + ⅔\, dt^2\, a(t) - ⅙\, dt^2\, a\,(t - dt)$
$v(t + dt) = v(t) + ⅓\, dt\, a(t + dt) + ⅚\, dt\, a(t) - ⅙\, dt\, a(t - dt)$
Aquest algorisme té una millor conservació de l'energia que l'original.
Totes aquestes modificacions generen la mateixa trajectòria que l'algorisme de Verlet original:
$r(t + dt) = 2r(t) - r(t - dt) + dt^2 a(t)$

Abast

L'objectiu del projecte és en un primer pas aplicar l'algorisme de Verlet a diversos sistemes físics i després proposar una modificació d'aquest. Els sistemes físics seran els següents:

1. Un objecte amb una altura inicial i una velocitat inicial aleatòries sota l'efecte del camp gravitatori. S'implementarà amb python perquè és un llenguatge de scripting que permet un desenvolupament molt ràpid.
2. Una massa m unida a una molla totalment elàstica de constant recuperadora k (el que se sol anomenar l'oscil·lador harmònic lliure) es pot descriure amb les equacions de moviment de l'oscilador harmonic simple. S'implementarà una interfície d'interacció persona computador en la que es mostrarà una animació de les molles. La interfície permeterà canviar els algoritmes a utilitzar i els paràmetres d'aquests mitjançant checkboxes pels algoritmes i widgets d'entrada de text per als paràmetres. També s'implementarà la possibilitat de canviar tant els algoritmes com els paràmetres mitjançant un sistema de comandes a través d'un petit llenguatge de programació. S'utilitzarà aprenentatge automàtic per corregir l'entrada en cas que l'usuari cometi errors.
3. Un sistema de partícules. S'utilitzarà un programa en Fortran proporcionat pel director del projecte. S'implementarà una modificació de l'algorisme de Verlet i gràfics 3D del sistema de partícules amb Fortran i OpenGL. Com que en aquest cas no existeix cap fòrmula que ens doni els resultats teòrics es mesuraran diverses propietats físiques del sistema i es compararan els resultats obtinguts amb els diversos algoritmes. Per mesurar les propietats físiques s'utilitzarà un altre programa en Fortran facilitat també pel director del projecte.

En el cas de l'objecte sota el camp gravitatori s'implementaran el resultats teorics, el mètode de Verlet i la modificació d'aquest. En el cas de l'oscilador harmònic s'implementarà a part d'aquest el mètode corrector predictor. El mètode corrector predictor no es pot aplicar en el cas de l'objecte sota el camp gravitatori degut a que l'acceleració és constant. En el cas del sistema de partícules s'implementaran el mètode de Verlet, la modificació i el mètode corrector predictor (no s'implementaran els resultats teòrics perquè en aquest cas no existeixen).
Tant en el cas de l'objecte sota l'efecte del camp gravitatori com en l'oscilador harmònic simple es realitzaran experiments de precisió per diferents valor de dt. En el cas del sistema de partícules es realitzaran experiments d'eficiència.
Els obstacles apareixeran en el sistema de partícules ja que hauré d'aprendre Fortran i buscar una llibreria que em permeti implementar els gràfics 3D. També hi

ha la possibilitat d'executar el programa en Fortran guardar els resultats en un fitxer i obrir-lo amb C++ i implementar els gràfics 3D amb C++.

Metodologia i rigor

Es farà servir una metodologia àgil.
Primer s'implementarà l'objecte sota el camp gravitatori amb els resultats teòrics i els resultats d'aplicar l'algorisme de Verlet.
Després s'implementarà l'oscil·lador harmònic simpleamb les seves equacions pròpies
Tot seguit es proposarà una modificació de l'algorisme de Verlet i s'aplicarà als 3 sistemes. Un cop trobada la modificació es realitzaran experiments d'eficiència i precisió (eficiencia al sistema de partícules i precisió als altres dos).
Després s'implementaran els gràfics 3D del sistema de partícules.
Per acabar s'implementarà la interfície d'interacció usuari computador, el llenguatge de programació per canviar els algoritmes i els paràmetres i el corrector d'aquest llenguatge mitjançant aprenentatge automàtic.
S'utilitzarà el llenguatge Python pels casos de l'objecte sota el camp gravitatori i l'oscil·lador harmònic simple perquè és un llenguatge de scripting que permet agilitzar la programació a cost de perdre eficiencia això però no ens preocupa perquè per aquests dos casos es faran proves de precisió.
El director del projecte proporcionarà un programa en Fortran del sistema de partícules per tant la modificació de l'algorisme de Verlet en aquest sistema s'implementarà també en Fortran.
Per implementar els gràfics 3D del sistema de partícules s'ha executat el programa en Fortran, s'ha guardat el resultat en un fitxer de text, s'ha obert des de C++ i s'ha implementat amb OpenGL.

Planificació temporal

Aquest projecte s'inicia el dia 16 de setembre de 2019 i està previst finalitzar-lo el 10 de gener de 2020. La duració total és d'unes 300 hores i la data prevista per la lectura és el torn de gener de 2020. Cada dia es treballen 3 hores.
Aquest treball es fa a la universitat i no té recursos materials.

Descripció de les tasques

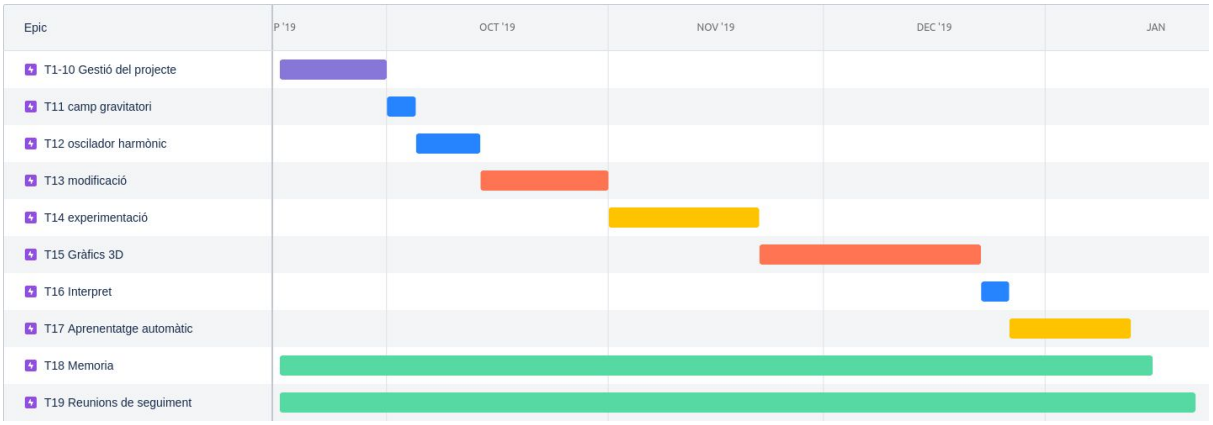
- Gestió del projecte(38 hores)
    - Contextualització (4 hores)
    - Justificació (4 hores)
    - Abast (4 hores)
    - Metodologia i rigor (2 hora)
    - Referències (2 hora)
    - Planificació temporal (7 hores)
        - Descripció de les tasques (3 hores)
        - Estimacions i Gant (3 hores)
        - Gestió del risc (1 hora)
    - Pressupost (4 hores)
    - Informe de sostenibilitat (4 hores)
- Objecte sota el camp gravitatori (10 hores): S'implementarà un sistema físic de la trajectòria d'un objecte sota el camp gravitatori per les equacions teòriques i el mètode Verlet leap frog.
- Oscil·lador harmònic simple (20 hores): S'implementarà un sistema físic d'una molla amb les equacions de l'oscil·lador harmònic simple per les equacions teòriques, el mètode Verlet leap frog i el mètode corrector predictor.
- Proposar una modificació de l'algorisme de Verlet (40 hores): es proposarà una o varies modificacions de l'algorisme de Verlet i s'implementarà en els 3 sistemes físics.
- Experimentació (40 hores): es faran proves d'eficiència en el cas del sistema de partícules i de precisió en els altres 2.
- Gràfics 3D del sistema de partícules (50 hores)
- Compilador del llenguatge de programació de la interfície d'interacció persona computador (10 hores)
- Aprenentatge automàtic(40 hores): s'utilitzarà aprenentatge automàtic per corregir l'entrada de l'usuari del llenguatge de programació en cas que sigui incorrecte.
- Memoria (80 hores): S'escriurà a mida que es progressi en el treball.
- Reunions de seguiment (6 hores): Es faran reunions de seguiment amb el professor que dirigeix el projecte aproximadament cada dues setmanes per mostrar el progrés fet.

Codi	Nom	Temps estimat	Dependencies
T1	Contextualització	4 hores	-
T2	Justificació	4 hores	-
T3	Abast	4 hores	-
T4	Metodologia i rigor	2 hora	-
T5	Referències	2 hora	T1, T2, T3, T4
T6	Descripció de les tasques	3 hores	-
T7	Estimacions i Gant	3 hores	T6
T8	Gestió del risc	1 hora	T6
T9	Pressupost	4 hores	
T10	Informe de sostenibilitat	3 hores	-
T11	Objecte sota el camp gravitatori	10 hores	-
T12	Oscilador harmònic simple	20 hores	-
T13	Modificar l'algorisme de Verlet	40 hores	T11, T12
T14	Experimentació	40 hores	T11, T12, T13
T15	Gràfics 3D	50 hores	-
T16	Interpret	10 hores	T12, T13
T17	Aprenentatge automàtic	40 hores	T16
T18	Memoria	80 hores	T10, T11, T12, T13, T14, T15, T16,T17
T19	Reunions de seguiment	6 hores	T11, T12, T14

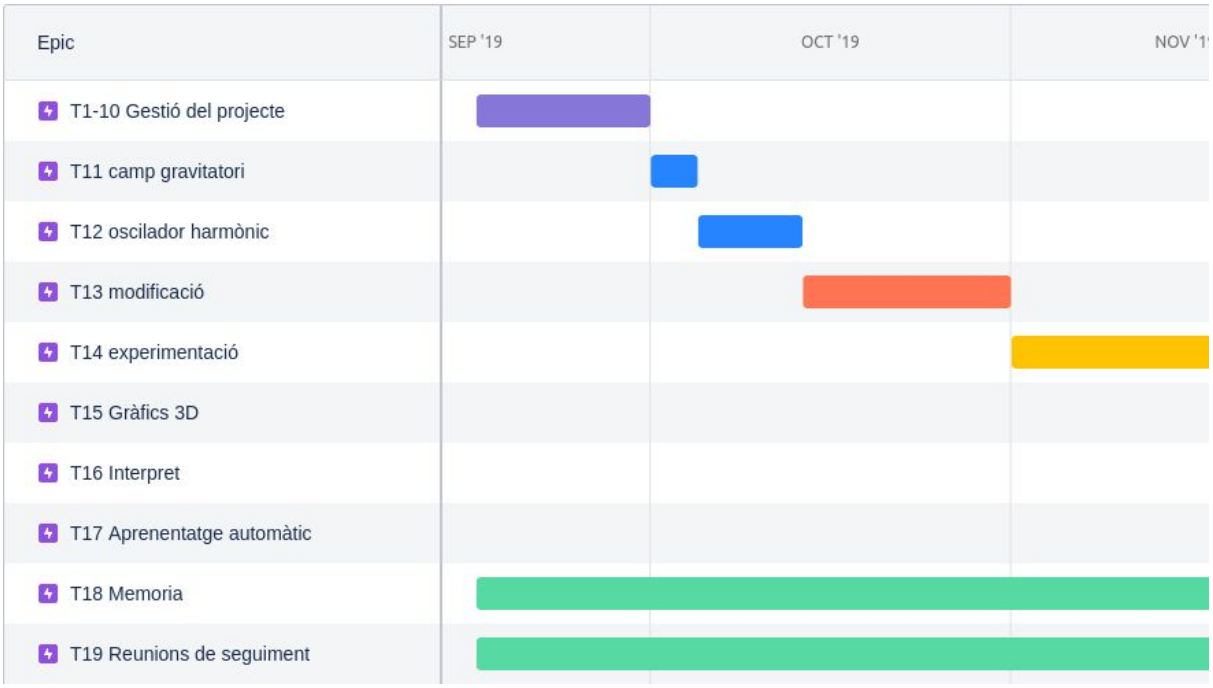
Plans alternatius i obstacles

Els obstacles poden ser trigar més del previst en qualsevol de les tasques, en aquest cas s'aumentaran el número d'hores dedicades.
Un altre obstacle és implementar els gràfics 3D del sistema de partícules ja que el programa està en Fortran com a pla alternatiu es guardarà la sortida del programa en un fitxer i s'implementaran els gràfics 3D en C++ i OpenGL . El motiu d'això és que Fortran disposa de poques llibreries i informació.
Un altre obstacle és millorar l'eficiència o precisió modificant l'algorisme de Verlet. En cas que no s'aconsegueixi es crearà un altre algorisme igual de precís i eficient partint d'un altre esquema com els mètodes lineals de múltiples pasos.

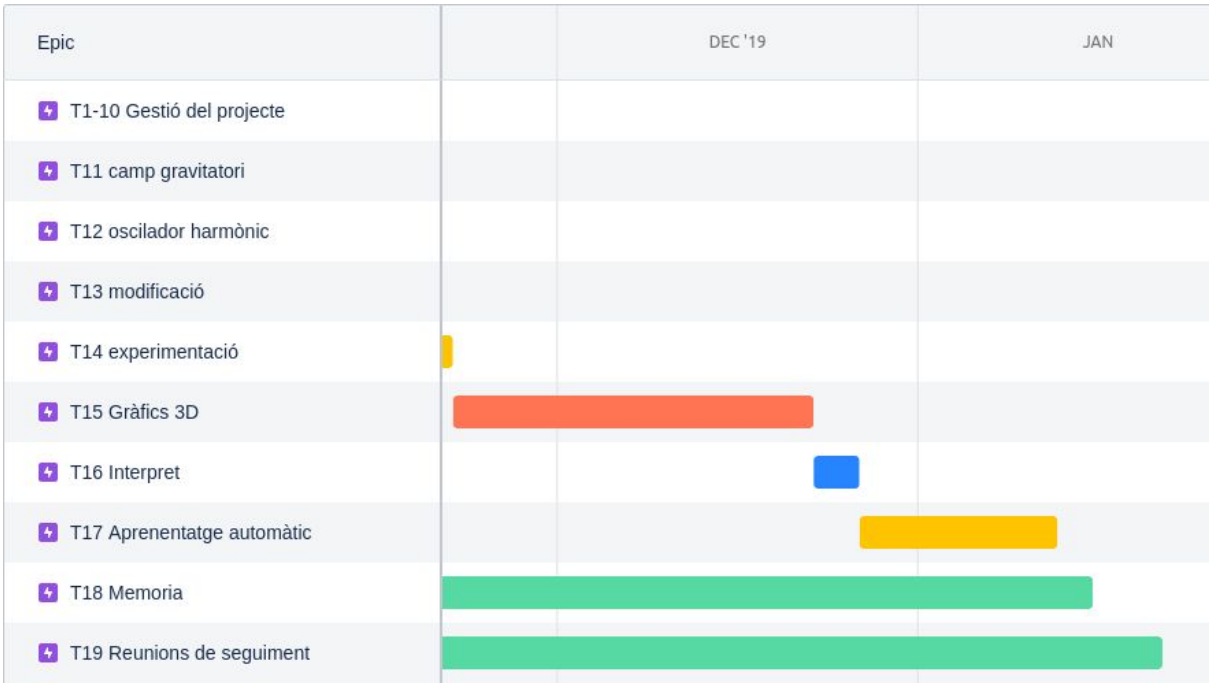
Diagrama de Gantt



Ampliat primera part



Ampliat segona part



Pressupost

Identificació i estimació dels costos

Hi ha 217 laborals a l'anys.
Els costos de recursos humans son els següents:

- Pel que fa a la gestió del projecte és un treball de gestor de projectes i el salari segons la font és d'uns 26000 euros a l'any. Això equival a uns 20,22 euros per hora, incloent la seguretat social. (26000 / 217 / 8 * 1.35)
- Les tasques de l'objecte sota el camp gravitatori, l'oscil·lador harmònic, la modificació de l'algorisme de Verlet, l'experimentació, els gràfics 3D, l'intèrpret i l'aprenentatge automàtic són treball d'un enginyer informàtic i el salari segons la font és d'uns 20000 euros a l'any. Això equival a uns 15,55 euros per hora, incloent la seguretat social. (20000 / 217 / 8 * 1.35)
- Les tasques de seguiment es fan entre un cap de projecte, el gestor de projectes i el programador junior. El cost del programador junior i del gestor de projectes és el mateix d'abans. El director de projecte cobra uns 36000 euros a l'any segons la font. Això equival a uns 28 euros l'hora, incloent la seguretat social. (36000 / 217 / 8 * 1.35). Tot suma 63 euros l'hora.
- Pel que fa a les tasques de redacció de la memoria l'escriurien entre el programador junior i el gestor de projectes com que el cost mig dels dos és d'uns 17,5 euros. ((20,22 + 14,78)/2)

Els costos generics es redueixen a dos ordinadors, un per el gestor de projectes i un per el programador junior i l'electricitat que consumeixen aquests, ja que tot el software utilitzat es open source.Cada ordinador costa uns 500 euros.
L'ordinador gasta uns 220 W per hora mentres està encès.
El preu del kw/h es d'uns 0,127 euros.
Com que la durada del projecte és d'unes 300 hores el cost en energia és de 66 kW/h (0,22 * 300) i això costa 8, euros (66 * 0,127).

	Unitat (%)	Preu unitat(euros)	Info unitat	Temps(hores)	Total(euros)	Observacions
Gestio del projecte				30,00	606,57	
Contextualització	0,058	35100,00	sou anual gestor de projectes	4,00	80,88	Estimació per unitats de recursos
Justificació	0,058	35100,00	sou anual gestor de projectes	4,00	80,88	Estimació per unitats de recursos
Abast	0,058	35100,00	sou anual gestor de projectes	4,00	80,88	Estimació per unitats de recursos
Metodologia i rigor	0,058	35100,00	sou anual gestor de projectes	2,00	40,44	Estimació per unitats de recursos
Referencies	0,058	35100,00	sou anual gestor de projectes	2,00	40,44	Estimació per unitats de recursos
Planificació temporal	0,058	35100,00	sou anual gestor de projectes	7,00	141,53	Estimació per unitats de recursos
Pressupost	0,058	35101,00	sou anual gestor de projectes	3,00	60,66	Estimació per unitats de recursos
Informe de sostenibilitat	0,058	35100,00	sou anual gestor de projectes	4,00	80,88	Estimació per unitats de recursos
Implementació				170,00	3266,13	Estimació per unitats de recursos
Camp gravitatori	0,058	27000,00	sou anual d'un enginyer informàtic	10,00	155,53	Estimació per unitats de recursos
Oscilador harmònic	0,058	27000,00	sou anual d'un enginyer informàtic	20,00	311,06	Estimació per unitats de recursos
Modificació	0,058	27000,00	sou anual d'un enginyer informàtic	40,00	622,12	Estimació per unitats de recursos
Experimentació	0,058	27000,00	sou anual d'un enginyer informàtic	40,00	622,12	Estimació per unitats de recursos
Gràfics 3D	0,058	27000,00	sou anual d'un enginyer informàtic	50,00	777,65	Estimació per unitats de recursos
Interpret	0,058	27000,00	sou anual d'un enginyer informàtic	10,00	155,53	Estimació per unitats de recursos
Aprenentatge automàtic	0,058	27000,00	sou anual d'un enginyer informàtic	40,00	622,12	Estimació per unitats de recursos
Reunions de seguiment	0,058	110700,00	sou anual del cap de projectes, el programador junior i el gestor	6,00	382,60	Estimació per unitats de recursos
Memoria	0,058	31050,00	sou anual del programador junior i del gestor de projectes	80,00	1430,88	Estimació per unitats de recursos
Total CPA					5686,18	
Ordinador	1,000	500,00	Un ordinador	2 unitats	1000,00	Estimació per unitats de recursos
Electricitat	1,000	0,13	Un kw/h	340,00	43,18	Estimació per unitats de recursos
Total CG					1043,18	
TotalCG + CPA					6729,36	
Contingencia	15,00%				1009,40	
Total CG + CPA + Contingencia					7738,76	
No acabar la gestio del projecte a temps (Cost = 123, Prob = 10%)					12,3	
No acabar la implementació a temps (Cost 502, Prob = 20%)					100,4	
No acabar a temps la memoria (cost 280, Prob = 10%)					28	
Total Imprevistos					140,7	
Total					7879,46	

Control de Gestió

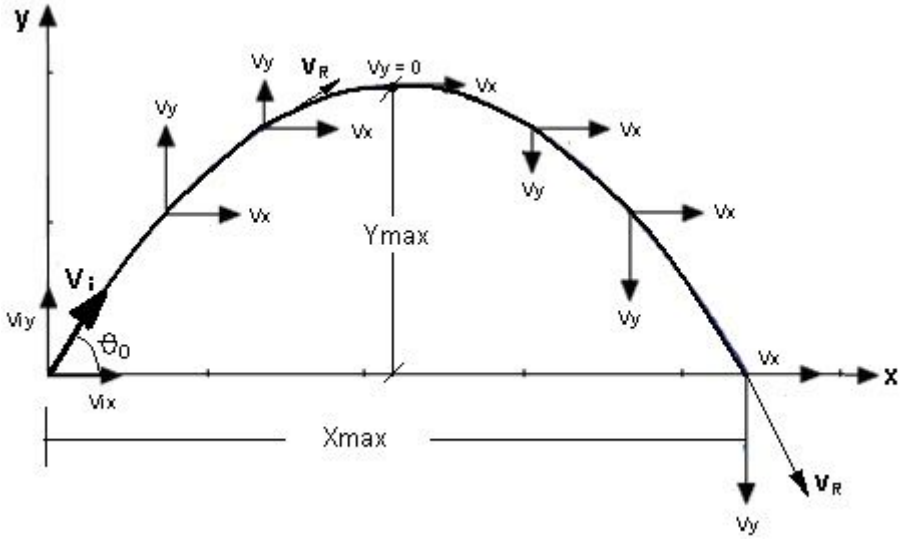
Es compararan les desviacions amb els costos planejats. Es controlarà en quina tasca s'ha produït la desviació, la justificació d'aquesta i la seva magnitud. Com a indicadors de control s'utilitzarà la desviació total aquesta és la diferència entre el cost real i el cost predit del projecte.

Sistemes físics

S'han implementat 3 sistemes, el camp gravitatori, l'oscil·lador harmònic simple i un sistema de partícules.

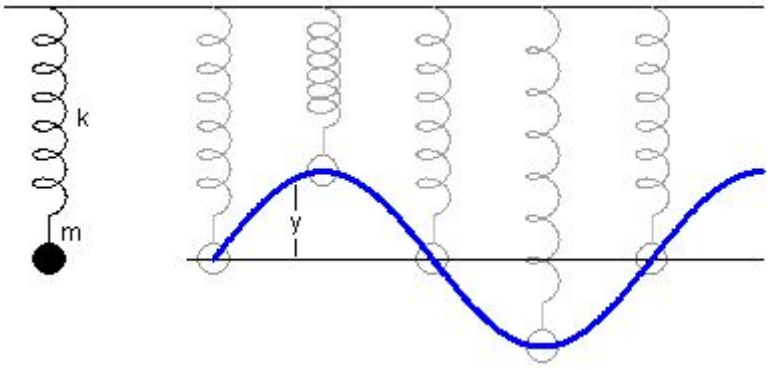
Camp gravitatori

Consisteix en un objecte amb velocitat i posició inicials aleatòries sota l'efecte del camp gravitatori. En aquest sistema coneixem el valor exacte. S'utilitzarà per fer proves de precisió.



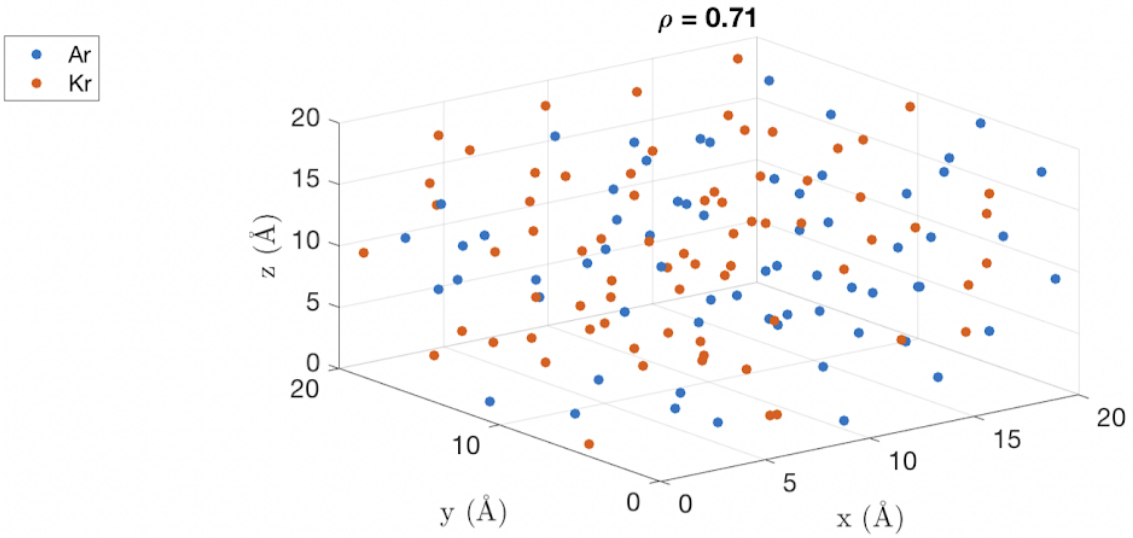
Oscilador harmònic simple

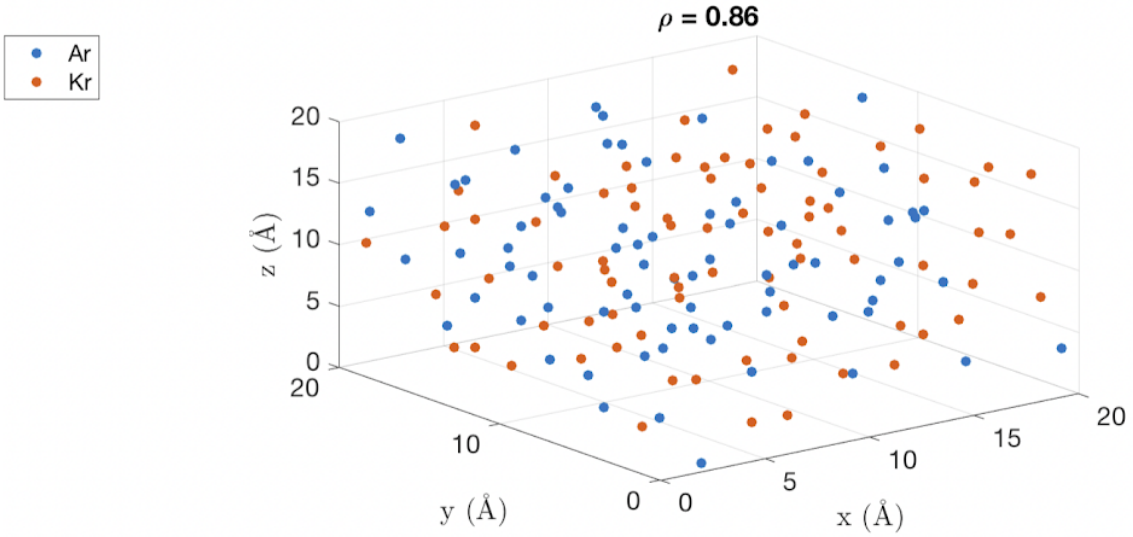
Consisteix en una massa unida a una molla elàstica que es comporta amb les equacions del moviment harmònic simple. La molla té una amplitud (A),una freqüència angular (w) i una fase inicial (fase0) aleatoris. En aquest sistema coneixem també el valor exacte. S'utilitzarà per fer proves de precisió.



Sistema de partícules

Consisteix en un model atòmic que representa un sistema d'àtoms d'Argon en forma líquida. Es construeix amb una caixa caixa de costat L i N partícules a dintre tal que quan una partícula surt per un costat, n'entra una altra idèntica pel costat contrari. Aquesta forma d'implementar un sistema infinit s'anomena aplicar "condicions periòdiques de contorn" i equival a considerar còpies (rèpliques) exactes de la capsa principal al llarg de les 3 direccions de l'espai. Aquesta prescripció és un recurs de simulació totalment estandarditzat en les simulacions de sistemes atòmics i moleculars. En aquest sistema no coneixem el valor exacte, només coneixem els resultats experimentals. S'utilitzarà per fer proves d'eficiència.





Anàlisi d'alternatives

S'ha intentat modificar l'algorisme de Verlet calculant r(t + 2dt) a partir de r(t + dt), r(t), r(t - dt) i r(t - 2dt), v(t) i a(t) d'aquesta forma es podria calcular l'acceleració cada 2dt en comptes de cada dt. Això seria beneficiós perquè en els sistemes de partícules el càlcul més costos és el de les forces (a partir de les quals es calculen les acceleracions).
De forma similar a l'algorisme de Verlet:
$r(t + 2dt) = r(t) + v(t)(2dt) + ½\ a(t)(2dt)^2 + ⅛\ a'(t)(2dt)^3 + 1/24\ a''(t)(2dt)^4 =$
$= r(t) + 2\ v(t)\ dt + 2\ a(t)\ dt^2 + 8/6\ a'(t)\ dt^3 + 16/24\ a''(t)\ dt^4$
$r(t + dt) = r(t) + v(t)\ dt + ½\ a(t)\ dt^2 + ⅛\ a'(t)\ dt^3 + 1/24\ a''(t)\ dt^4$
$r(t - dt) = r(t) - v(t)\ dt + ½\ a(t)\ dt^2 - ⅛\ a'(t)\ dt^3 + 1/24\ a''(t)\ dt^4$
$r(t - 2dt) = r(t) - 2\ v(t)\ dt + 2\ a(t)\ dt^2 - 8/6\ a'(t)\ dt^3 + 16/24\ a''(t)\ dt^4$
Al sumar les 4:
$r(t + 2dt) = 4\ r(t) - r(t + dt) - r(t - dt) - r(t - 2\ dt) + 5\ a(t)\ dt^2 + 34/24\ a''(t)\ dt^4$
El problema és que l'algorisme de Verlet és més precís:
$r(t + 2dt) = r(t) + 2v(t)\ dt + 2\ a(t)\ dt^2 + 8/6\ a'(t)\ dt^3 + 16/24\ a''(t)\ dt^4$
$r(t - 2dt) = r(t) - 2\ v(t)\ dt + 2a(t)\ dt^2 - 8/6\ a'(t)\ dt^3 + 16/24\ a''(t)\ dt^4$
Al sumar les 2:
$r(t + 2dt) = 2\ r(t) - r(t - 2dt) + 4a(t)\ dt^2 + 32/24\ a''(t)\ dt^4$
$34/24\ a''(t)\ dt^4 > 32/24\ a''(t)\ dt^4$
Com que es suprimeix aquest terme l'algorisme de Verlet és més precís. Per aquest motiu s'ha descartat aquesta alternativa.

Alternativa escollida

S'ha obtat doncs pels mètodes lineals de múltiples pasos. Concretament pels mètodes de Adams-Bashforth perquè els mètodes de Adams-Moulton requereixen l'acceleració a l'instant t per calcular la posició al mateix instant t però per calcular l'acceleració és necessària la posició. Tot i que es podria utilitzar un mètode corrector predictor això implicaría calcular l'acceleració 2 vegades i en els sistemes de partícules el càlcul de l'acceleració és molt costós.
S'ha adaptat els mètodes de Adams-Bashforth per calcular la velocitat a partir de les s acceleracions anteriors i la posició a partir de les s velocitats anteriors:
$v(t + dt) = v(t) + dt\ *(b\_s\text{-}1\ *\ a(t) + b\_s\text{-}2\ *\ a(t - dt) + \ldots + b\_0\ *\ a(t - s + 1))$
$r(t + dt) = r(t) + dt\ *\ (b\_s\text{-}1\ *\ v(t) + b\_s\text{-}2\ *\ v(t - dt) + \ldots + b\_0\ *\ v(t - s + 1))$
Els mètodes originals només calculen la el nou valor a partir dels valors previs de la primera derivada. Jo l'he aplicat recursivament per trobar la primera derivada a partir de la segona.

Mètodes lineals de múltiples pasos

Tenim el valor de la derivada de la funció que volem aproximar i un punt inicial:

$y' = f(t, y), \quad y(t_0) = y_0.$

El mètodes lineals de múltiples pasos utilitzen una combinació lineal de y_i i $f(t_i, y_i)$ per calcular el següent valor de y :

$$\begin{aligned} & y_{n+s} + a_{s-1} \cdot y_{n+s-1} + a_{s-2} \cdot y_{n+s-2} + \cdots + a_0 \cdot y_n \\ & \quad = h \cdot (b_s \cdot f(t_{n+s}, y_{n+s}) + b_{s-1} \cdot f(t_{n+s-1}, y_{n+s-1}) + \cdots + b_0 \cdot f(t_n, y_n)) \\ & \Leftrightarrow \sum_{j=0}^{s} a_j y_{n+j} = h \sum_{j=0}^{s} b_j f(t_{n+j}, y_{n+j}), \end{aligned}$$

Els mètodes poden ser explícits o implícits: si $b_s = 0$ el mètode és explicit, es pot calcular directament. Si $b_s \neq 0$ el mètode és implícit, y_{n+s} depen de $f(t_{n+s}, y_{n+s})$.

Adams-Bashforth methods

Són mètodes explícits. Els coeficients $a_{s-1} = -1, \; a_{s-2} = ... = a_0 = 0$

Els coeficients b_j es determinen mitjançant interpolació polinòmica per trobar el polinomi p de grau s - 1 tal que:

$p(t_{n+i}) = f(t_{n+i}, y_{n+i}), \qquad \text{for } i = 0, \ldots, s-1.$

Mitjançant la fórmula de Lagrange obtenim

$$p(t) = \sum_{j=0}^{s-1} \frac{(-1)^{s-j-1} f(t_{n+j}, y_{n+j})}{j!(s-j-1)!h^{s-1}} \prod_{\substack{i=0 \\ i \neq j}}^{s-1} (t - t_{n+i}).$$

El polinomi p és una bona aproximació local de f(t,y) en l'equació diferencial y'=f(t, y) així podem utilitzar l'equació y' = p(t), ja que aquesta equació la podem resoldre exactament. Podem utilitzar la següent equació:

$$y_{n+s} = y_{n+s-1} + \int_{t_{n+s-1}}^{t_{n+s}} p(t)\, dt.$$

El metode d'Adams-Bashforth s'obte al substituir p(t).

Els coeficients b_j s'obtenen de la següent forma:

$$b_{s-j-1} = \frac{(-1)^j}{j!(s-j-1)!} \int_0^1 \prod_{\substack{i=0 \\ i \neq j}}^{s-1} (u+i)\, du, \qquad \text{for } j = 0, \ldots, s-1.$$

Substituir f(t,y) per el polinomi interpolador té un error de h^s . Els mètodes d'Adams-Bashforth tenen ordre s.

Adams–Moulton methods

Els mètodes d'Adams-Moulton són implícits, ja que no tenen la restricció que $b_s = 0$. Els coeficiens a_j són iguals que en els mètodes d'Adams-Bashforth $a_{s-1} = -1,\ a_{s-2} = ... = a_0 = 0$.
Els mètodes d'Adams-Moulton s'obtenen igual que els mètodes d'Adams-Bashforth tret que el polinomi interpolador utilitza el punt t_n a part dels punts $t_{n-1}, ..., t_{n-s}$.
Els coeficients b_j s'obtenen amb la següent fórmula:

$$b_{s-j} = \frac{(-1)^j}{j!(s-j)!} \int_0^1 \prod_{\substack{i=0 \\ i \neq j}}^{s} (u+i-1)\, du, \qquad \text{for } j = 0, \ldots, s.$$

Per resoldre l'equació implícita es pot utilitzar el mètode de Newton o el mètode predictor-corrector.

Mètode predictor-corrector

Són algorismes per resoldre equacions diferencials. Consten 2 pasos:

1. Predicció. A partir dels valor d'una funció i les seves derivades en un conjunt de punts anterior predir el valor de la funció en un nou punt.
2. Correcció. Refina l'aproximació utilitzant el valor de la funció predit en el pas anterior i la seva derivada.

Utilitzen un mètode explicit per la predicció i un mètode implícit per la correcció. Com que aquest mètode necesita calcular la derivada de la funció 2 cops cada iteració i el càlcul de les forces és la part més costosa en els sistemes de partícules utilitzar el mètode d'Adams-Moulton amb un mètode corrector predictor no és adequat.

Mètode de Newton

És un algorisme per trobar 0s en funcions continues. Produeix successivament millors aproximacions als 0s de les funcions.
Funciona iterant la següent fórmula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

El problema es que a cada iteració s'ha de tornar a calcular la derivada i en els sistemes de partícules el càlcul de les forces és la part més costosa. Per això utilitzar el mètode d'Adams-Moulton amb el mètode de Newton no és adequat

Polinomi de Lagrange

Els polinomis de Lagrange s'utilitzen per interpolació polinòmica en anàlisi numèric. Per un conjunt de punts (x_j, y_j) sense 2 x_j iguals el polinomi de Lagrange és el polinomi de grau més petit que passa per tots els punts.
Donat un conjunt de k + 1 punts:

$$(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$$

El polinomi de Lagrange és:

$$L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$$

On:

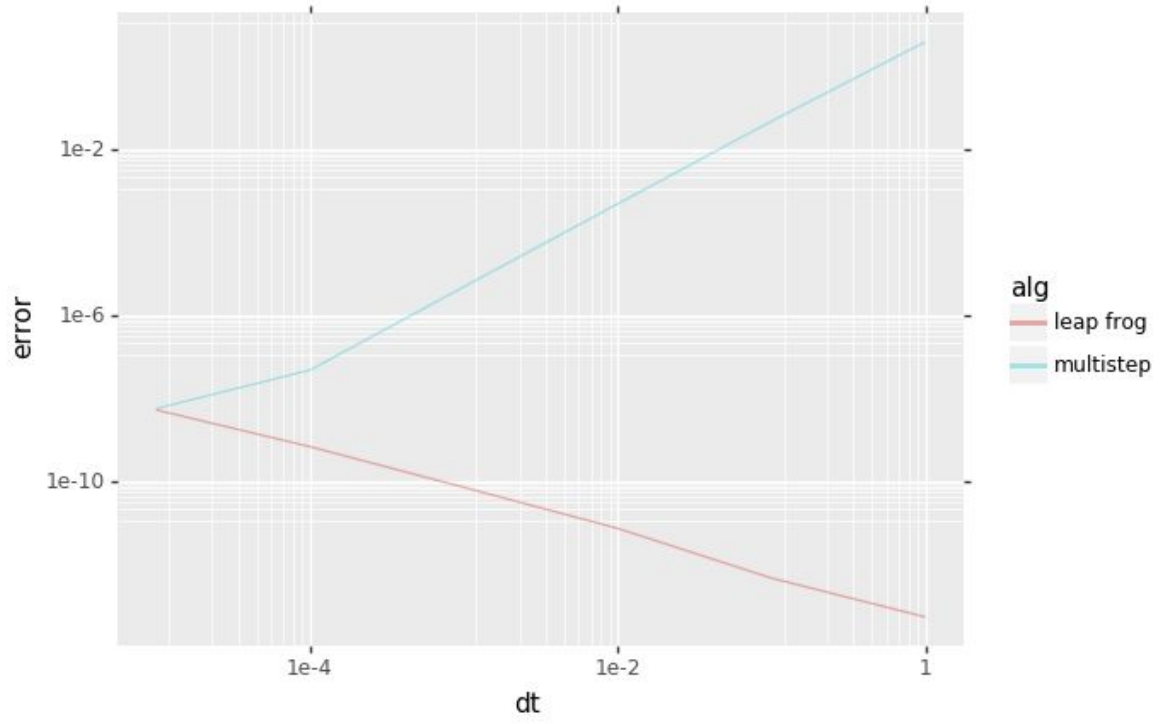
$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)},$$

Complexitat

Analitzarem la complexitat de l'algorisme sense el càlcul de propietats físiques.
Sigui P el número de partícules el càlcul de les posicions de l'algorisme de Verlet leap-frog té complexitat O(P) perquè itera per totes les partícules i les 3 components de cada partícula (x, y, z). Sigui s el número de passos utilitzats en el mètode multistep, aquest té complexitat O(s P) perquè itera per tots els passos anteriors i totes les partícules i les 3 components de cada partícula.
Pel que fa al càlcul de les forces té complexitat $O(P^2)$ perquè crida a la funció lj per cada parell d'àtoms i aquesta té complexitat constant ja que només realitza operacions constants.
LLavors l'algorisme de Verlet leap-frog té cost temporal $O(P^2 + P) = O(P^2)$ per iteració i el mètode multistep té cost temporal $O(P^2 + sP) = O(P^2)$ per iteració. Perquè P és més gran que s. Sigui I el número d'iteracions el cost total d'ambdós algorismes és el mateix $O(IP^2)$.
Pel que fa a la complexitat espacial de l'algorisme de Verlet leap-frog, aquest guarda la acceleració, la velocitat i la posició, el cost es $O(P)$. El mètode multistep guarda la posició anterior i l'acceleració i velocitat de les s posicions anteriors, el cost és doncs $O(sP)$.
Per tant si s és una constant comparada amb P (s és molt menor que P) llavors els dos algorismes tenen la mateixa complexitat espacial $O(P)$.

Experimentació

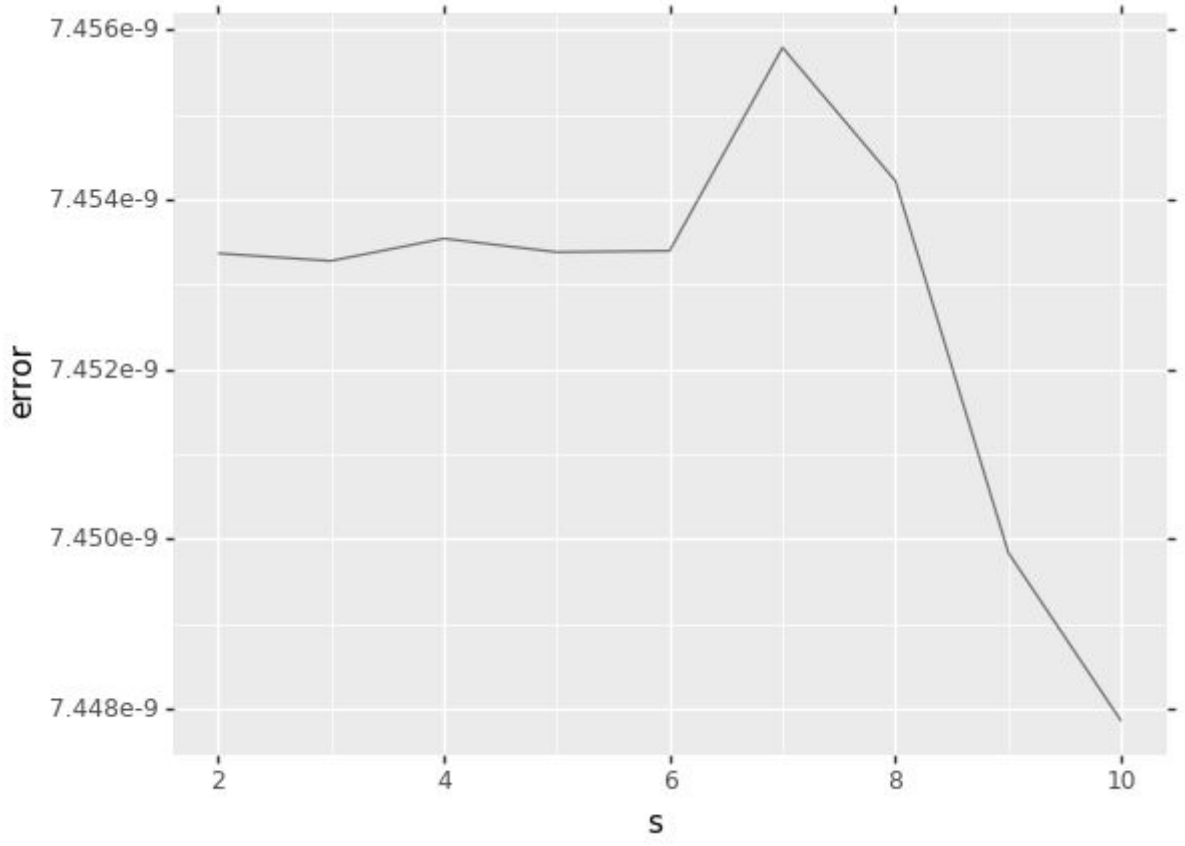
Observació	Els dos algoritmes (leap frog i multistep) poden tenir diferent precisió segons dt en el camp gravitatori.
Plantejament	Mesurem la precisió dels dos algoritmes per diferents valors de dt.
Hipòtesis	Els dos algoritmes són iguals independentment del valor de dt (H0) o tenen diferent precisió segons el valor de dt.
Mètode	• Generarem 10 problemes aleatoris (generant aleatòriament els valors de y0, v0x i v0y). • Mesurarem l'error dels dos mètodes per cada problema aleatori i per cada valor de dt (1,1e-1,1e-2,1e-3,1e-4,1e-5) i calculem l'esperança i la desviació estàndard de l'error respecte els resultats teòrics per cada valor de dt dels 10 problemes aleatoris.



dt	esperance leap frog	desviacio leap frog	esperance multistep	desviacio multistep
1	5,42854E-14	4,20004E-14	3,92	1,82120E-13
0,1	4,69391E-13	3,79562E-13	0,04802	3,89607E-13
0,01	7,40713E-12	7,76327E-12	0,00048902	9,89477E-12
0,001	7,22658E-11	7,22007E-11	0,000004899	9,40324E-11
0,0001	6,89961E-10	5,53009E-10	4,83471E-08	6,02120E-10
0,00001	5,30820E-09	0,000000004	5,69707E-09	4,10031E-09

Conclusió: Per a valor de dt inferiors a 1e-4 és evident que l'algoritme leap frog té menys error que el multistep.
Per els valor de dt 1e-4 i 1e-5 assumirem una distribució normal i realitzarem un test d'hipòtesis per veure si la mitjana de les distribucions és identica. Utilitzarem un test t-Student:
dt = 1e-4
Ttest_indResult(statistic=-184.34026447050368, pvalue=9.272031785157886e-31)
Com podem veure la probabilitat que siguin iguals és molt petita per tant per 1e-4 < dt < 1 l'algorisme leap frog és més precís.
dt = 1e-5
Ttest_indResult(statistic=-0.2157655554957468, pvalue=0.8315995900896503)
La probabilitat que siguin iguals és alta i per tant els dos algoritmes són igual de precisos per dt <= 1e-5

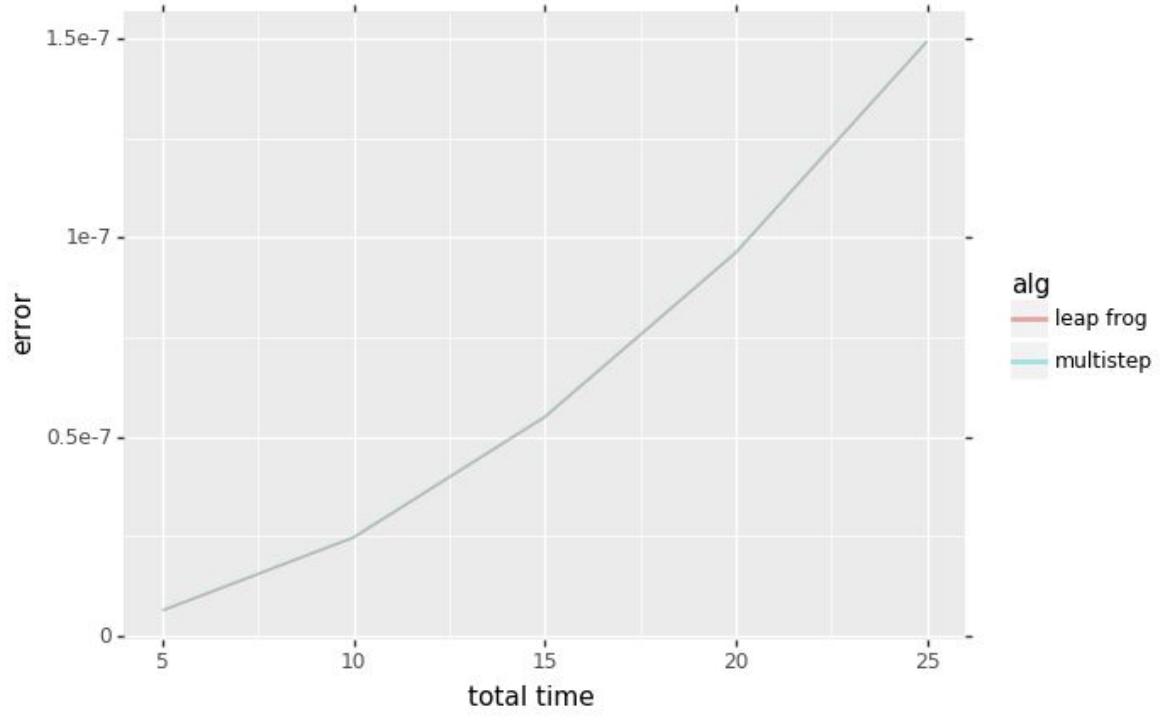
Observació	Diferents valors de s poden tenir diferent precisió (multistep) en el camp gravitatori.
Plantejament	Mesurem la precisió de l'algorisme multistep per diferents valors de s.
Hipòtesis	La precisió de l'algorisme multistep és independent de s (H0) o depen de s.
Mètode	● Generarem 10 problemes aleatoris (generant aleatòriament els valors de y0, v0x i v0y). ● Mesurarem l'error de l'algorisme multistep per cada problema aleatori i per cada valor de s (1,2,3 …,9,10) i calculem l'esperança i la desviació estàndard de l'error respecte als valors teòrics per cada valor de s dels 10 problemes aleatoris.



s	**esperance**	**desviació**
2	7,45336E-09	2,993142E-09
3	7,45327E-09	2,993116E-09
4	7,45353E-09	2,993226E-09
5	7,45337E-09	2,993152E-09
6	7,45339E-09	2,993168E-09
7	7,45579E-09	2,993777E-09
8	7,45421E-09	2,993443E-09
9	7,44983E-09	2,992341E-09
10	7,44784E-09	2,991844E-09

Conclusió: Per a valors de s entre 2 i 10 la precisió del mètode multistep és igual.

Observació	Diferents valors del temps total poden tenir diferent precisió per als dos algorismes (leap frog i multistep) en el camp gravitatori.
Plantejament	Mesurem la precisió dels dos algorismes per diferents valors del temps total.
Hipòtesis	La precisió dels dos algorismes no depèn del temps total (H0) o si que hi depèn.
Mètode	● Generarem 10 problemes aleatoris (generant aleatòriament els valors de y0, v0x i v0y). ● Mesurarem l'error dels tres algorismes per cada problema i per cada valor del temps total (5,10,15,20,25) i calculem l'esperança i la desviació estàndard de l'error respecte els resultats teòrics dels 10 problemes aleatoris per cada valor del temps total.



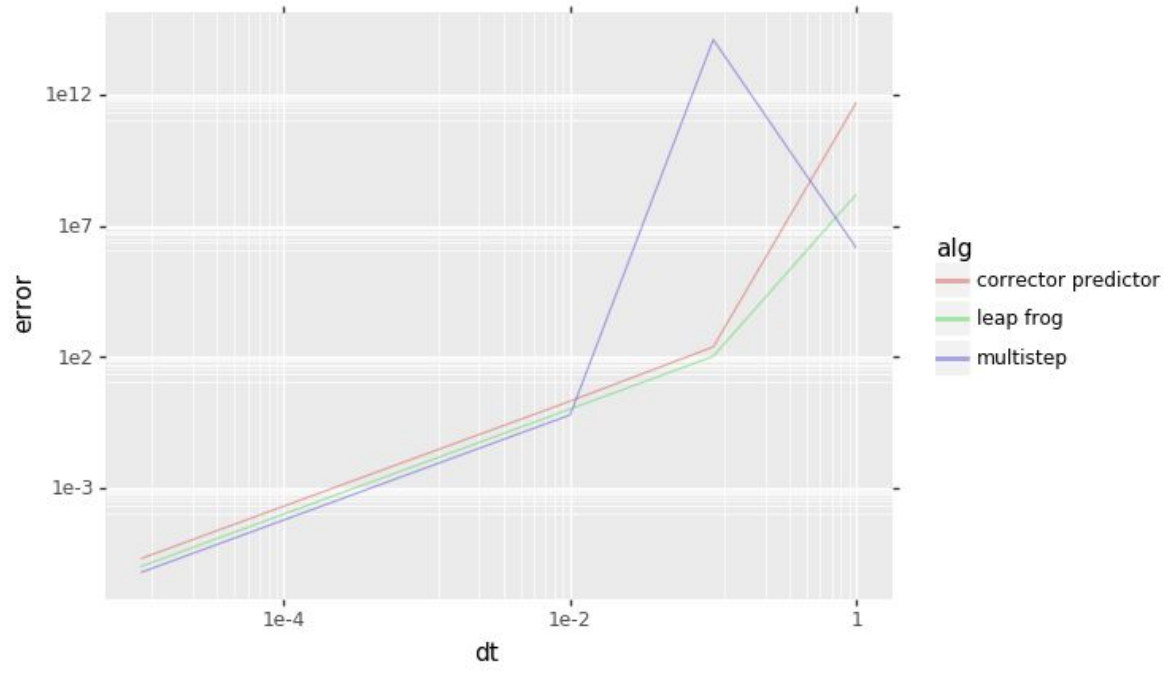
temps total	**esperance leap frog**	**desviacio leap frog**	**esperance multistep**	**desviacio multistep**
5	6,28422106640246E-09	3,43307436729515E-09	6,56713420045038E-09	3,74497052188615E-09
10	2,46132120901143E-08	1,43274777907709E-08	2,49277540993253E-08	1,45895290391703E-08
15	5,48645528975711E-08	3,27823749627128E-08	5,52047591447295E-08	0,000000033
20	9,61572102148199E-08	6,04481743053766E-08	9,65102765543885E-08	6,06627347183446E-08
25	1,49204346208714E-07	9,61026092338941E-08	1,49528281662535E-07	9,63666055441843E-08

Conclusió: Per tots els valors assumirem una distribució normal i realitzarem un test d'hipòtesis per veure si la mitjana de les distribucions és identica. Utilitzarem un test t-Student:

temps total	statistic	pvalue
5	-0.17609715709557966	0.8621983298567003
10	-0.04864323891745355	0.961739464665601
15	-0.02312326245170354	0.9818063748058797
20	-0.013037291244650417	0.9897414591831434
25	-0.007526824635589965	0.9940773139929153

Com podem veure la probabilitat que els dos mètodes siguin igual de precisos és molt alta.

Observació	Els tres algoritmes (leap frog, multistep i corrector predictor) poden tenir diferent precisió segons dt en l'oscil·lador harmònic simple.
Plantejament	Mesurem la precisió dels tres algoritmes per diferents valors de dt.
Hipòtesis	Els tres algoritmes són iguals independentment del valor de dt (H0) o tenen diferent precisió segons el valor de dt.
Mètode	• Generarem 10 problemes aleatoris (generant aleatòriament els valors de w, A i fase0). • Mesurarem l'error dels tres metodes per cada problema aleatori i per cada valor de dt (1,1e-1,1e-2,1e-3,1e-4,1e-5) i calculem l'esperança i la variança de l'error dels 10 problemes aleatoris per cada valor de dt.



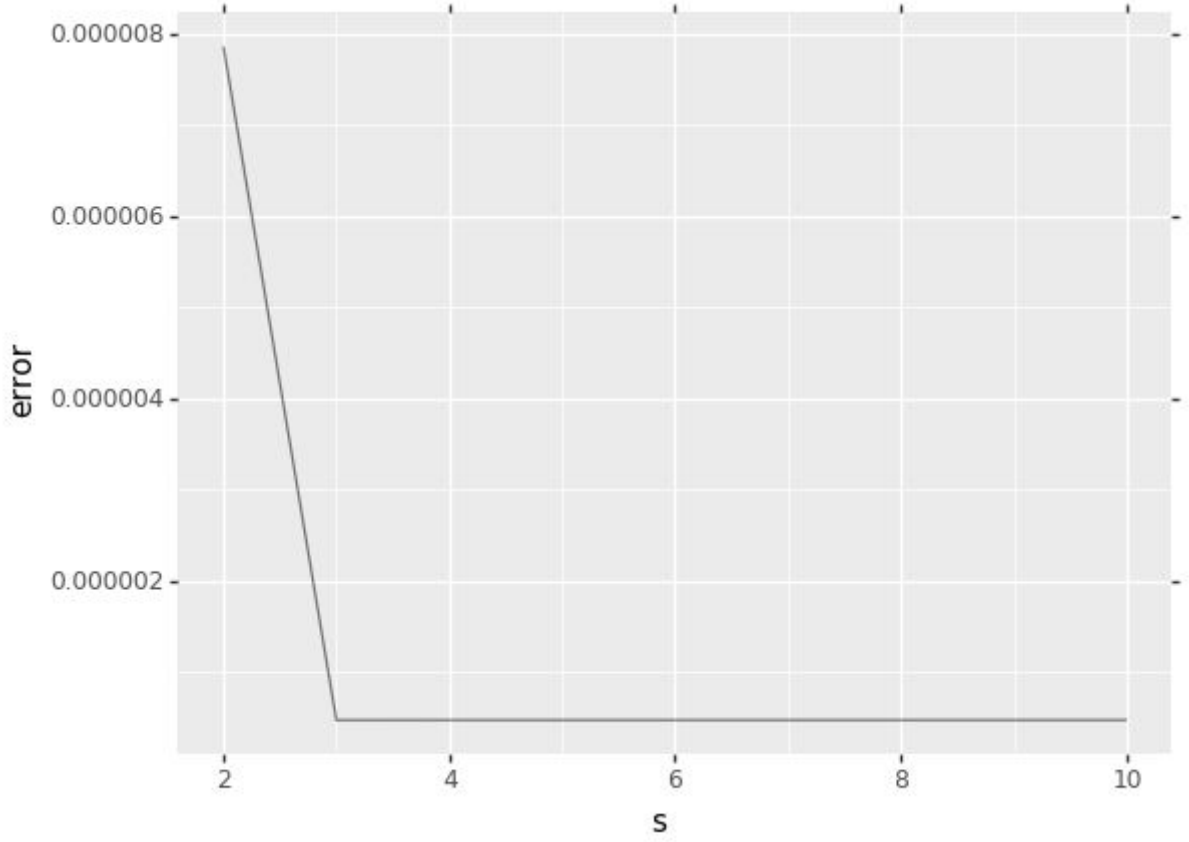
dt	esperance leap frog	desviation leap frog	esperance multistep	desviation multistep	esperance corrector predictor	desviation corrector predictor
1	158822198	274266993	1432167	2050963	501303910693	780782111339
0,1	107,847409	117,511290	123812746008692	337454722236094	246,2700280	283,2378534
0,01	1,05967349	1,15437732	0,631348922	0,575768007	2,108711619	2,296753629
0,001	0,01062862	0,01157933	0,006322353	0,005771152	0,021251174	0,023152505
0,0001	0,00010632	0,00011584	6,323158E-05	5,771859E-05	0,000212649	0,000231676
1e-05	1,0633E-06	1,1584E-06	6,323310E-07	5,772017E-07	2,126633E-06	2,316915E-06

Conclusió: Els valors de dt = 1 i dt = 0.1 són casos degenerats, pels altres valors assumirem una distribució normal i realitzarem un test d'hipòtesis per veure si la mitjana de les distribucions és identica (entre leap frog i multistep). Utilitzarem un test t-Student:

dt	statistic	pvalue
0.01	1.0499867634471292	0.3125447694676229
0.001	1.052544135135791	0.3114210693821632
1e-4	1.0529872467226458	0.31122837243615775
1e-5	1.053008609566273	0.3112190908813446

Per tots els valors la probabilitat que siguin iguals és baixa per tant podem concloure que el mètode multistep és més precís que el mètode verlet leap frog.

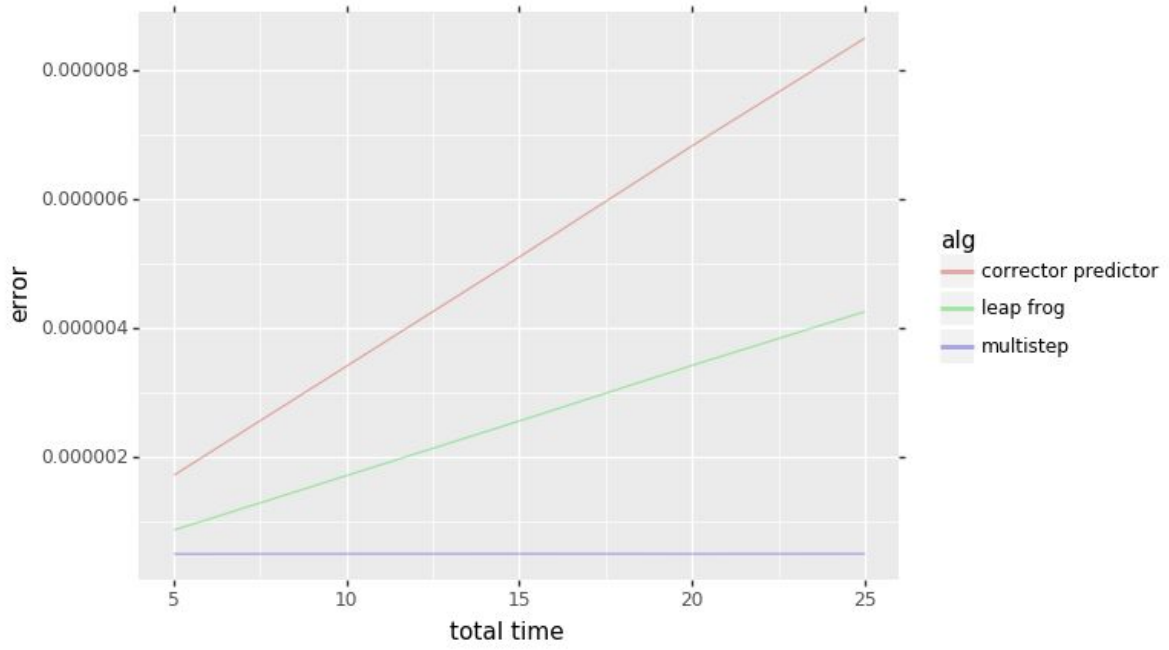
Observació	Diferents valors de s poden tenir diferent precisió (multistep) en l'oscil·lador harmònic simple..
Plantejament	Mesurem la precisió de l'algorisme multistep per diferents valors de s.
Hipòtesis	La precisió de l'algorisme multistep és independent de s (H0) o depen de s.
Mètode	● Generarem 10 problemes aleatoris (generant aleatòriament els valors de w, A i fase0). ● Mesurarem l'error de l'algorisme multistep per cada problema aleatori i per cada valor de s (1,2,3 …,9,10) i calculem l'esperança i la variança de cada valor de s per als 10 problemes aleatoris.



s	error esperance	error deviation
2	1,37E-5	1,34E-5
3	7,97E-7	6,92E-7
4	7,98E-7	6,93E-7
5	7,98E-7	6,93E-7
6	7,98E-7	6,93E-7
7	7,98E-7	6,93E-7
8	7,98E-7	6,93E-7
9	7,98E-7	6,93E-7
10	7,98E-7	6,93E-7

Conclusió: En el cas de l'oscil·lador harmònic pel valor de s = 2 el mètode multistep és menys precís.

Observació	Diferents valors del temps total poden tenir diferent precisió per als tres algorismes (leap frog, multistep i corrector predictor) en l'oscil·lador harmònic simple.
Plantejament	Mesurem la precisió dels tres algorismes per diferents valors del temps total.
Hipòtesis	La precisió dels tres algorismes no depèn del temps total (H0) o si que hi depèn.
Mètode	● Generarem 10 problemes aleatoris (generant aleatòriament els valors de w, A i fase0). ● Mesurarem l'error dels tres algorismes per cada problema i per cada valor del temps total (5,10,15,...45,50) i calculem l'esperança i la variança per cada valor del temps total dels 10 problemes aleatoris.



total time	esperance leap frog	desviation leap frog	esperance multistep	desviation multistep	esperance corrector predictor	desviation corrector predictor
5	8,5325E-07	1,6212E-06	4,831086E-07	8,161513E-07	1,706512E-06	3,242471E-06
10	1,7011E-06	3,2245E-06	4,851866E-07	8,211778E-07	3,402306E-06	6,449125E-06
15	2,5494E-06	4,8294E-06	4,858643E-07	8,228009E-07	5,098794E-06	9,658884E-06
20	3,4132E-06	6,4704E-06	0,000000485	8,212288E-07	6,826413E-06	0,000012941
25	4,2496E-06	8,0462E-06	0,000000486	8,231592E-07	8,499246E-06	1,609246E-05

Conclusió: Per tots els valors assumirem una distribució normal i realitzarem un test d'hipòtesis per veure si la mitjana de les distribucions (entre leap frog i multistep) és identica. Utilitzarem un test t-Student:

total time	statistic	pvalue
5	0.6448820633372477	0.5299818762843858
10	1.1555965219898516	0.2742925081212342
15	1.3319913497261033	0.21386418823655995
20	1.4196869813067998	0.1883730680949218
25	1.471480147863315	0.17456239648241845

Podem veure que a mesura que augmenta el temps de simulació disminueix la probabilitat que els dos mètodes siguin igual de precisos, per tant el mètode multistep conserva millor la precisió.

Observació	Els 3 algorismes poden tenir diferent eficiencia.
Plantejament	Mesurarem el temps total dels 3 algorismes.
Hipòtesis	Els 3 algorismes són igual d'eficients (H0) o són diferents en eficiencia.
Mètode	• Executarem els 3 algorismes en els sistema de partícules amb 100000 iteracions i mesurarem el temps que triguen.

	verlet leap frog	multistep	corrector predictor
100000 iteracions	2879 s	2960 s	4428s
1 iteració	28.79 ms	29.6 ms	44.28ms

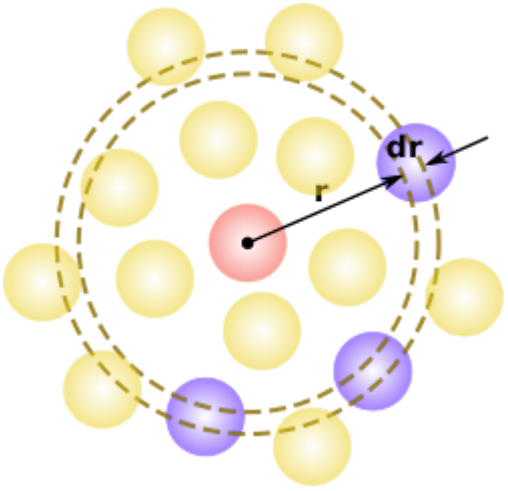
Podem veure que el més ineficient és el mètode corrector predictor això era esperable ja que ha de calcular les forces 2 cops per cada iteració. El mètode multistep és més ineficient que el verlet leap frog però no massa.

Conclusió general

En el cas del camp gravitatori per un diferencial de temps suficientment petit els dos mètodes (multistep i leap frog) són igual de precisos. En l'oscil.lador harmònic simple en canvi el mètode multistep és més precís. Podem concloure que depèn del sistema l'algorisme multistep pot ser més precís que el leap frog. Pel que fa a l'eficiència és evident que l'algorisme multistep és més costós que el leap frog (perquè es fan més càlculs). Però en el cas que hi hagi altres càlculs més costosos a fer (com el càlcul de forces en el sistema de partícules) el cost de l'algorisme multistep esdevé similar al del leap frog.

Funció de distribució radial

Descriu com la densitat varia en funció de la distància respecte una partícula de referencia. És una mesura de la probabilitat de trobar una partícula a certa distancia de la partícula de diferencia.
L'algorisme determina quantes partícules hi ha entre les distàncies r i r + dr. A la imatge la partícula de referencia és la rosa, i les partícules lilas son les que el seu centre es troba entre les distàncies r i r + dr.



Sigui N el número de partícules, V el volum, T la temperatura, r les coordenades de cada partícula, β el factor de Boltzmann $\beta = \frac{1}{kT}$ i l'energia potencial deguda a la interacció entre partícules $U_N(\mathbf{r}_1 \ \ldots, \mathbf{r}_N)$.
Les mitjanes apropiades es prenen en la col·lectivitat canònica (N,V,T) amb la integral configuracional sobre totes les possibles combinacions de posicions de partícules.

$$Z_N = \int \cdots \int \mathrm{e}^{-\beta U_N} \, \mathrm{d}\mathbf{r}_1 \cdots \mathrm{d}\mathbf{r}_N$$

La probabilitat de trobar la partícula 1 en l'interval dr1, la partícula en l'interval dr2 … ve donada per:

$$P^{(N)}(\mathbf{r}_1, \ldots, \mathbf{r}_N) \, \mathrm{d}\mathbf{r}_1 \cdots \mathrm{d}\mathbf{r}_N = \frac{\mathrm{e}^{-\beta U_N}}{Z_N} \mathrm{d}\mathbf{r}_1 \cdots \mathrm{d}\mathbf{r}_N$$

Donat que el nombre de partícules és molt gran es pot obtenir la probabilitat d'una configuració reduïda on únicament n < N partícules estan fixades i no hi ha restriccions sobre les N - n partícules restants. Llavors cal integrar sobre les coordenades restants $\mathbf{r}_{n+1},\ldots,\mathbf{r}_N$:.

$$P^{(n)}(\mathbf{r}_1,\ldots,\mathbf{r}_n) = \frac{1}{Z_N}\int\cdots\int \mathrm{e}^{-\beta U_N}\,\mathrm{d}\mathbf{r}_{n+1}\cdots\mathrm{d}\mathbf{r}_N$$

Com que les partícules són idèntiques podem considerar la probabilitat que qualsevol partícula ocupi les posicions $\mathbf{r}_1,\ldots,\mathbf{r}_n$ en qualsevol permutació.

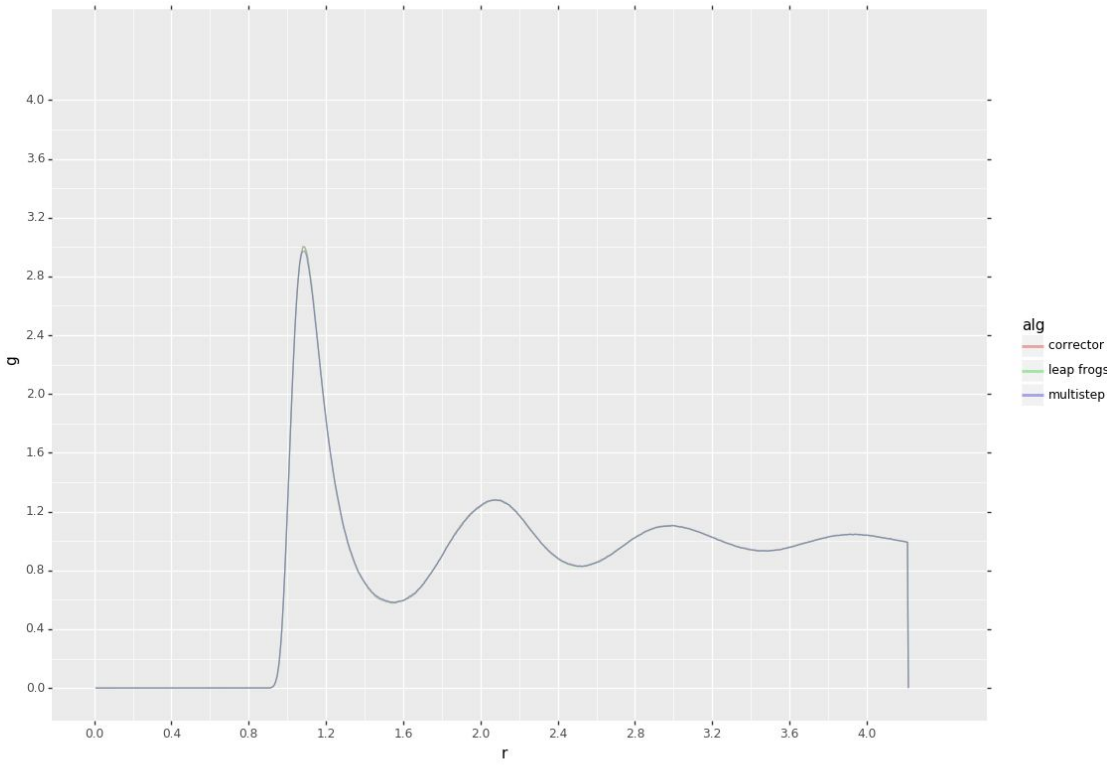
$$\rho^{(n)}(\mathbf{r}_1,\ldots,\mathbf{r}_n) = \frac{N!}{(N-n)!}P^{(n)}(\mathbf{r}_1,\ldots,\mathbf{r}_n)$$

La funció $g^{(n)}$ és la *funció de* correlació, si els àtoms són independents $\rho^{(n)}$ seria igual a ρ^n. $g^{(n)}$ corregeix la correlació entre àtoms.

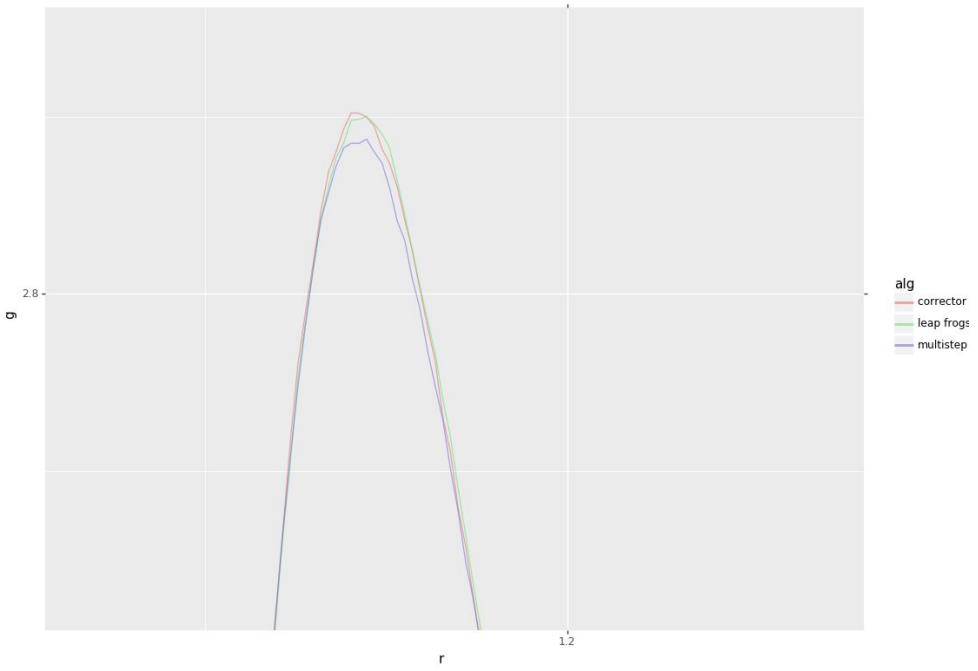
$$\rho^{(n)}(\mathbf{r}_1\ \ldots,\mathbf{r}_n) = \rho^n g^{(n)}(\mathbf{r}_1\ \ldots,\mathbf{r}_n)$$

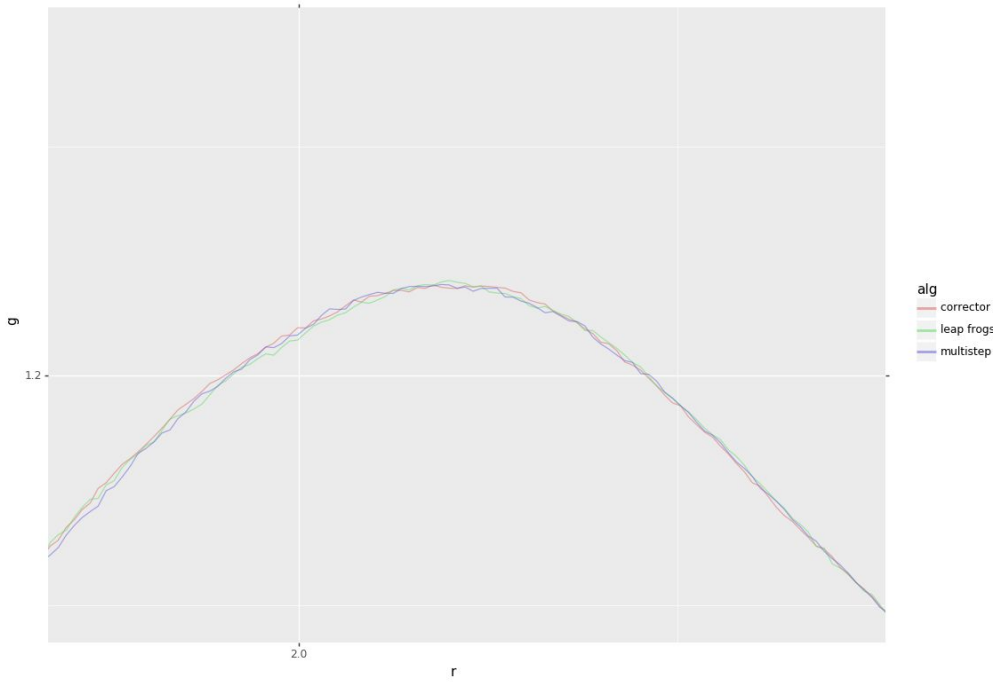
$$g^{(n)}(\mathbf{r}_1\ \ldots,\mathbf{r}_n) = \frac{V^n N!}{N^n(N-n)!}\cdot\frac{1}{Z_N}\int\cdots\int \mathrm{e}^{-\beta U_N}\,\mathrm{d}\mathbf{r}_{n+1}\cdots\mathrm{d}\mathbf{r}_N$$

Podem veure que la funció de distribució radial per als 3 algorismes és molt semblant.



Però si fem zoom podem veure diferències:





Difusió

La difusió és el procés a través del qual un sistema amb concentració no uniforme es torna uniforme per si sol. Així les molècules que es troben en una regió amb concentració més passen a una regió amb concentració menor.
Mitjançant la següent equació podem calcular la difusió:

$\frac{\partial <r^2(t)>}{\partial t}$ = 2dD

On r²(t) és el desplaçament quadràtic mitjà.
Amb la següent equació el podem calcular:

$$< \Delta r(t)^2 >= \frac{1}{N} \sum_{i=1}^{N} \Delta r_i(t)^2$$

On el desplaçament és la integral de la velocitat en la partícula.

$$\Delta r(t) = \int_0^t dt'\, v(t')$$

Hi una relació que expresa el coeficient de difusió directament a partir de les velocitats:

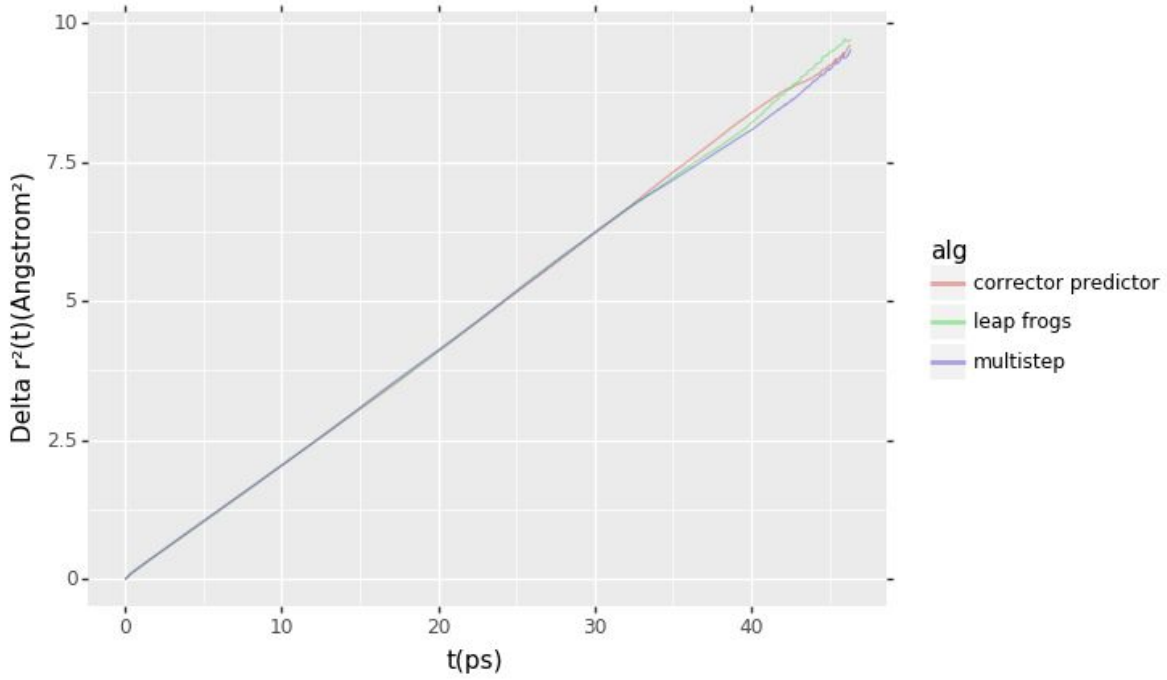
$$2D = \lim_{t\to\infty} \frac{\partial <x^2(t)>}{\partial t}$$

$$< x^2(t) >=< (\int_0^t dt'\, v_x(t'))^2 >= \int_0^t \int_0^t dt'dt'' < v_x(t')v_x(t'') >=$$

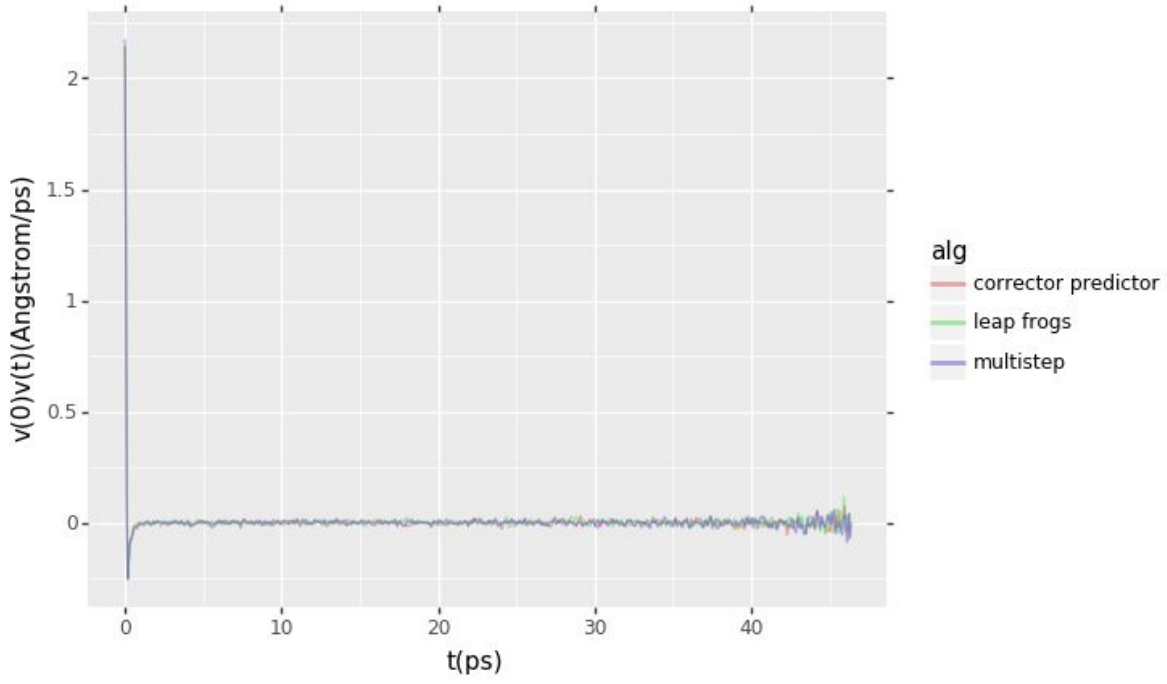
$$= 2\int_0^t \int_0^{t'} dt'dt'' < v_x(t')v_x(t'') >$$

$< v_x(t')v_x(t'') >$ és la funció d'autocorrelació de velocitats, mesura la correlació de les velocitats d'una partícula en els instants t' i t''.

Desplaçament quadràtic mitjà



Funció d'autocorrelació de velocitats



En els dos casos el tres algoritmes obtenen els mateixos valors al començament i difereixen amb el temps.

Implementació

Per representar els polinomis interpoladors en:

$$b_{s-j-1} = \frac{(-1)^j}{j!(s-j-1)!} \int_0^1 \prod_{\substack{i=0 \\ i \neq j}}^{s-1} (u+i)\, du, \qquad \text{for } j = 0, \ldots, s-1.$$

S'ha utilitzat una llista on la posició 0 representa el terme independent, la posició 1 el terme amb x, la posició 2 el terme amb x²…
Sumem els polinomis 2 a 2 sumant els elements de la mateixa posició:
Exemple:
poly_res[0] = poly1[0] + poly2[0]
…
poly_res[N] = poly1[N] + poly2[N]
Per multiplicar polinomis multipliquem un polinomi sencer per cada posició de l'altre i després sumem tots els polinomis resultants. Per multiplicar un polinomi sencer p1 per cada posició de l'altre p2[i] desplacem el resultat amb tants 0s com la posició de l'altre (i 0s) i després copiem el polinomi sencer p1 multiplicat per p2[i] començant per la posició i. (S'ha de tenir en compte que i comença per 0)
Per integrar polinomis realitzen l'integral de cada posició i acumulem els resultats. L'integral de cada posició és el coeficient de la posició dividit per la posició més 1. (

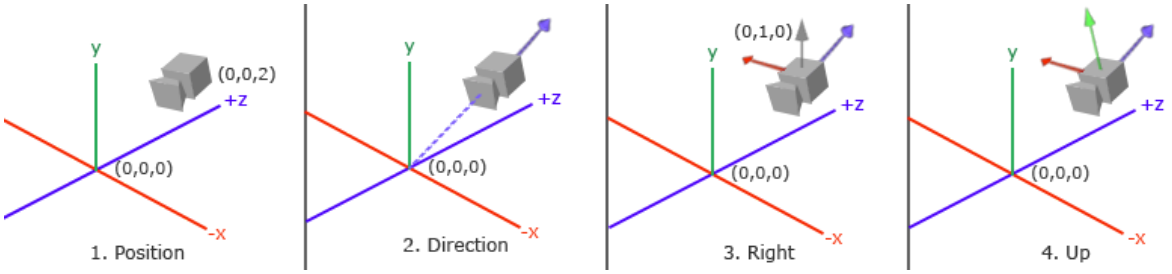
$$\int_0^1 cx^k = \frac{c1^{k+1}}{k+1} = \frac{c}{k+1}$$

Per calcular l'algorisme multistep, de grau s, primer es calculen els coeficients b per cada una de les s primeres posicions i després es calculen un cop per la resta de posicions (perquè el número de posicions anteriors i els coeficients b no varien).

Gràfics

Càmera

Per simular una càmera movem o rotem els objectes en la direcció oposada a la que es mou o rota la càmera.
Per definir la càmera necessitem la seva posició en coordenades de mon, la direcció cap a la que mira, un vector que apunti a la dreta i un que apunti cap a dalt.
El vector en la direcció cap a on mira la càmera ha d'anar desde l'objecte fins a la càmera i no de la càmera a l'objecte.
El vector que apunta a la dreta el podem calcular fent el producte vectorial entre el vector cap a on mira la càmera i un vector en la direcció de l'eix y.
Per calcular el vector que apunta cap a dalt fem un altre producte vectorial entre el vector cap a on mira la càmera i el vector que apunta a la dreta.



Look at

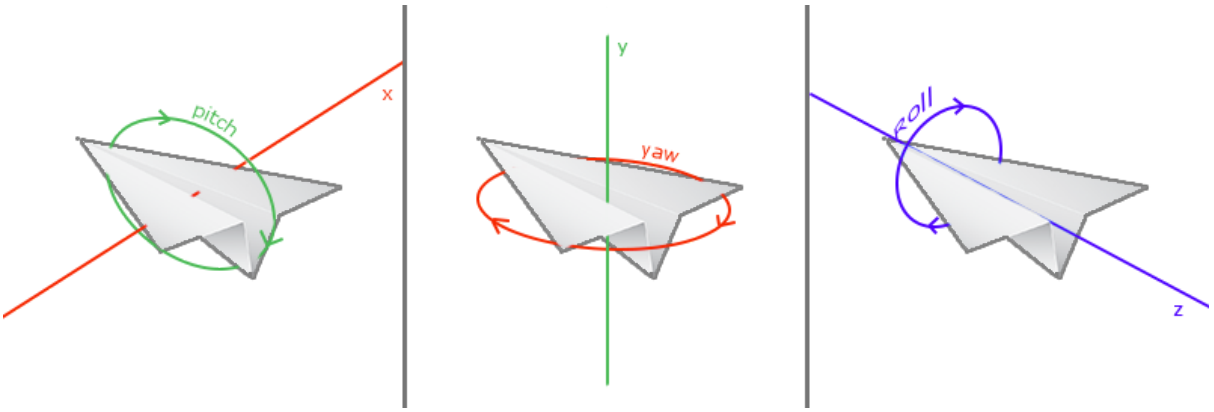
Si formem una matriu amb 3 vectors linealment independents i el vector de la translació podem transformar qualsevol vector a aquest sistema de coordenades multiplicant-lo per aquesta matriu.

$$\begin{bmatrix} right_x & up_x & forward_x & position_x \\ right_y & up_y & forward_y & position_y \\ right_z & up_z & forward_z & position_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

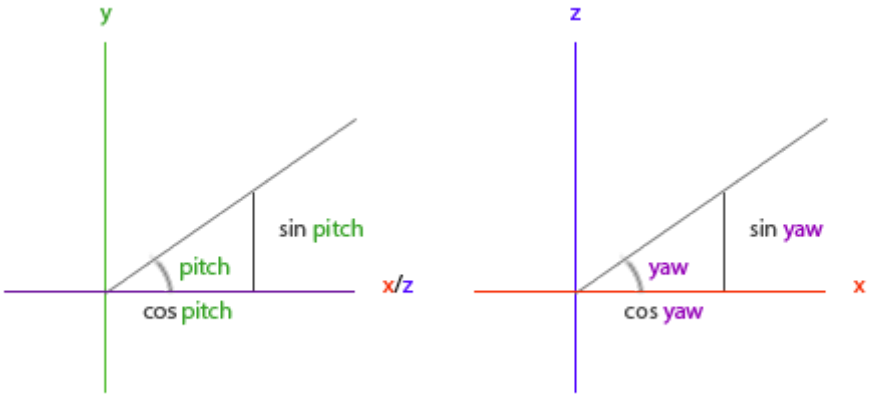
GLM té aquesta funció predefinida a partir de la posició la direcció de la càmera i el vector que apunta cap a dalt.

Euler angles

Són 3 variables que poden definir qualsevol rotació en 3D, roll, pitch i yaw.



En el nostre cas només utilitzarem pitch i yaw.



De la figura de l'esquerra podem deduir que::

- y = sin(pitch)
- x,z = cos(pitch)

De la figura de la dreta podem deduir que:

- x = cos(yaw)
- z = sin(yaw)

Per tant:

- x = cos(pitch) cos(yaw)
- y = sin(pitch)
- z = cos(pitch) sin(yaw)

Els valors de pitch i yaw els obtenim amb el moviment del ratolí on el moviment horitzontal afecta el yaw i el vertical el pitch.
S'ha fet servir una projecció perspectiva perquè és més realista que la ortogonal.
Tant per saber la posició del ratolí com per fer zoom GLFW disposa de callback functions que es criden quan succeeix un determinat event. Concretament `glfwSetCursorPosCallback` per la posició del ratolí i `glfwSetScrollCallback` per fer zoom.

Il·luminació

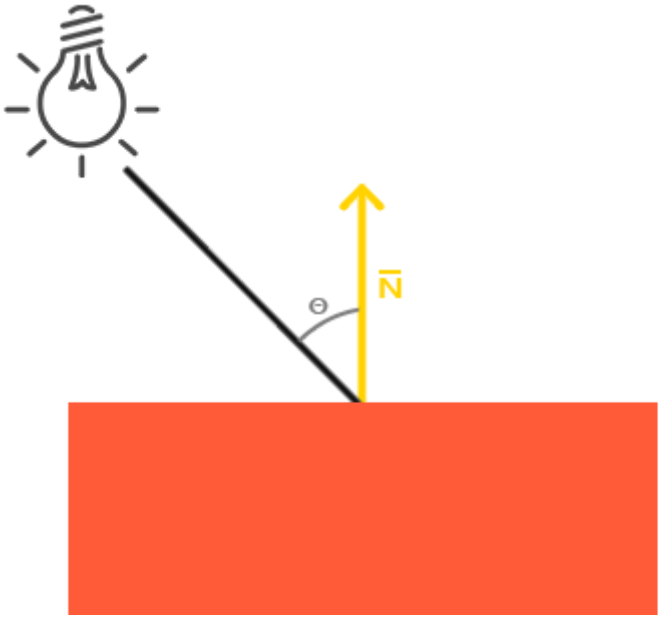
Utilitzem el model de Phong, consisteix en 3 components: ambient, difusa i especular.

Ambient

És una llum que prové de diferents fonts que poden no ser visibles. Es calcula multiplicant una constant (intensitat) pel color de llum.

Difusa

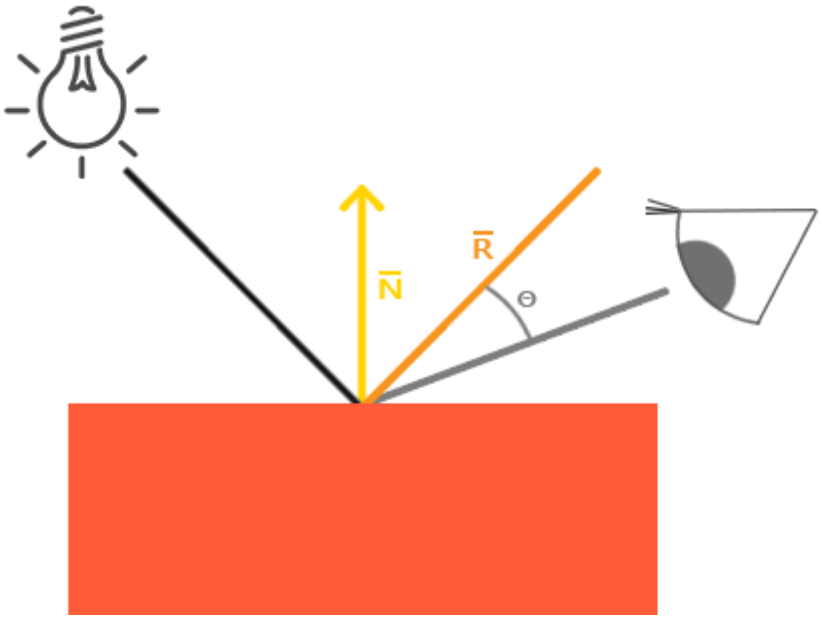
Dona als objectes més brillantor quan estan més alineats amb els rajos de llum.



Quan menor sigui l'angle entre la font de llum i la normal de la cara de l'objecte major impacte tindrà la font de llum. Es calcula al fragment shader mitjançant un producte escalar entre la normal del fragment i la direcció de la llum, ambdós en World Space. S'ha de vigilar que aquest producte escalar no sigui inferior a 0 i s'han de normalitzar els dos vectors.
Com que els vectors normals representen una direcció i no un punt en l'espai les translacions no haurien de tenir efecte sobre aquests, llavors hem d'agafar la matriu 3 x 3 esquerre superior per eliminar la translació. Si s'aplica un escalat no uniforme els vectors normals deixaran de ser perpendiculars a la superfície. Per solucionar això en comptes de la Model Matrix s'ha de d'utilitzar la transposada de la inversa de la Model Matrix.

Especular

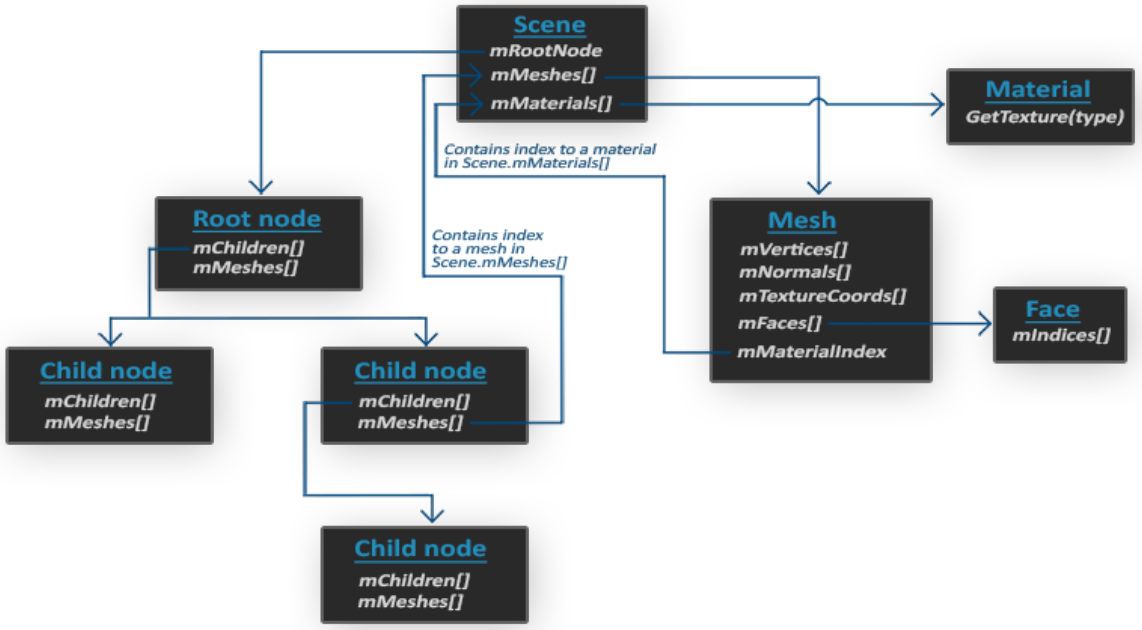
Depèn de la reflexió de la llum sobre l'objecte i la direcció de l'observador. Com més petit sigui l'angle entre la llum reflectida i l'observador major impacte tindrà la llum especular. També es calcula al fragment shader mitjançant un producte escalar entre el raig reflectit i la direcció de l'observador (També s'ha de vigilar que el resultat no sigui negatiu). El resultat s'eleva a una constant (brillantor), com major sigui aquesta l'objecte reflecteix més la llum i el punt es torna més petit.



Carregar el model

Assimp

És una llibreria que permet carregar els models en estructures de dades genèriques independentment del format del model.
Un exemple:



L'objecte Scene conté totes les dades del model o escena i un punter al root node. Un Mesh conté tota la informació necessària per dibuixar (vertexs, normals, coordenades de textura, cares (faces) i el material de l'objecte). Un Face representa una primitiva.

Mesh

Transforma les dades de les estructures de dades de Assimp en un format que OpenGL pugui tractar. Guarda tota la informació dels vèrtexs (posició, normal i coordenades de textura), els índexs i les textures. S'encarrega d'inicialitzar tots els buffers i atributs i de dibuixar l'objecte.
Els structs en C++ s'emmagatzemen en memoria de forma seqüencial. Així podem passar al buffer de OpenGL un vector de vèrtexs i especificar l'offset de cada atribut (posició, normal i coordenades de textura).
Com que no sabem quantes textures tindrà el model definirem la següent convenció: cada textura difusa serà texture_diffusei i cada textura especular texture_speculari on i és 1,...,n on n és el número de textures difuses o especulars. D'aquesta forma no importa quantes textures té el model i sabem el nom que tindran (necessitem saber el nom per programar el shader).

Model

Aquesta classe s'encarrega de carregar el model amb Assimp i transformar-lo en el número de Mesh que tingui, també s'encarrega de dibuixar tots els Mesh del model. Al carregar el model amb Assimp el triangula si no estava triangulat ja.
Per transformar l'objecte mesh de Assimp en el nostre accedim a les propietats que necessitem i les guardem en el nostre objecte. S'ha de processar les dades dels vertex, els índexs del mesh i les dades del material.
Pot haver moltes meshes que continguin la mateixa textura. Per explotar això podem guardar totes les textures en una variable global i al carregar una nova comprobar primer si no s'ha carregat ja.

Instancing

Serveix per dibuixar moltes instàncies d'un model amb diferents transformacions eficientment. En comptes d'una crida a glDrawArrays per cada instancia del teu model es crida una única vegada a glDrawArraysInstanced per totes les instàncies del teu model.
En el vertex shader hi ha un paràmetre predefinit adicional gl_InstanceID amb un valor diferent per cada instancia del model. Fent servir això podem passar un uniform amb les transformacions de totes les instàncies i accedir a la transformació de cada instància mitjançant el paràmetre gl_InstanceID.
Aquest mètode però no és escalable ja que hi ha un límit en la quantitat de dades que podem enviar en un uniform als shaders.
Una alternativa, anomenada instanced arrays, és definir les transformacions de cada instancia del model com un atribut del vertex (igual que la posició o el color).
Per tal de fer servir una transformació diferent amb cada instancia del model necessitem la funció glVertexAttribDivisor. Aquesta funció controla quan el contingut de l'atribut del vèrtex ha de passar al següent element mitjançant un paràmetre de la funció. Per defecte aquest paràmetre és 0 que significa que OpenGL actualitzarà el contingut de l'atribut del vèrtex cada iteració del Vertex Shader. Si val 1 el contingut de l'atribut del vèrtex s'actualitzarà quan dibuixem una nova instància.
La mida maxima dels atributs del vèrtex és vec4, com que la transformació és una mat4 hem de reservar 4 atributs de vec4 consecutius.

Interfície d’interacció persona computador

S’ha implementat una interfície que mostra els resultats de la simulació de l'oscil·lador harmònic simple. La interfície permet en una finestra seleccionar o deseleccionar els algorismes a utilitzar. Al enviar el resultat si no hi ha cap algorisme seleccionat sortirà un missatge d’error avisant d’això a l’usuari.
En una altra finestra permet canviar els paràmetres. Si al enviar l’usuari no ha entrat algun paràmetre se li preguntarà si vol generar-lo aleatòriament, si no vol generar-lo aleatòriament no s’aplicaran els valors introduïts i es mostrarà un missatge demanant a l’usuari que introdueixi el paràmetre.
En una tercera finestra l’usuari pot modificar els algorismes a utilitzar i el valor dels paràmetres mitjançant un petit llenguatge de comandes. El canvis realitzats a través del sistema de comandes es mostraran a les finestres dels paràmetres i algorismes. Hi haurà un botó d’ajuda que mostrarà un missatge sobre el format de les comandes.

Interpret

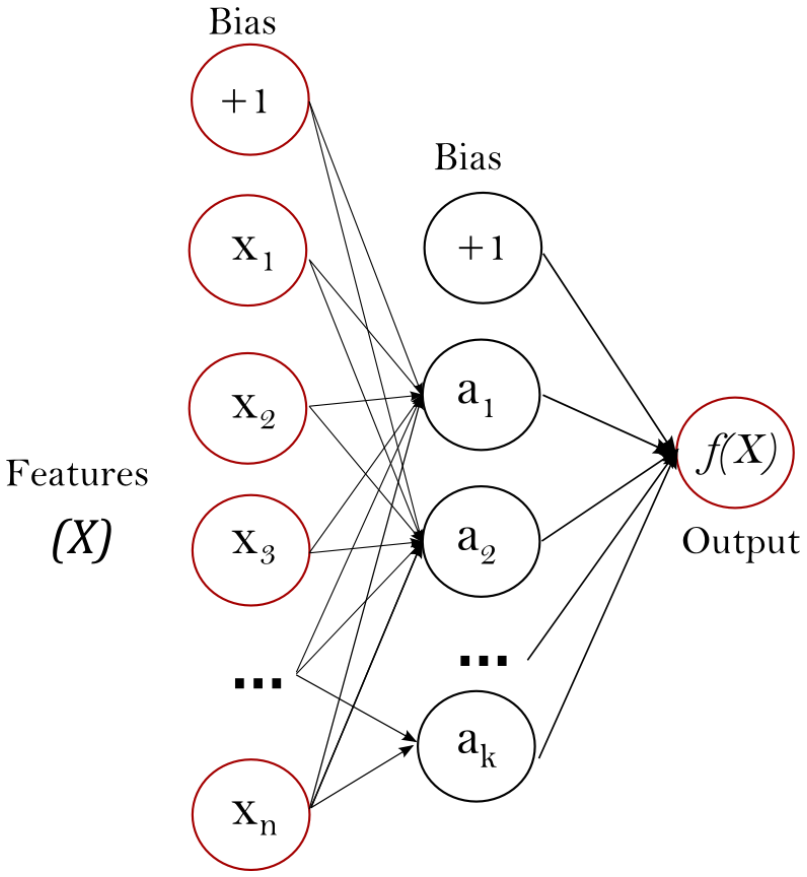
S’ha construït un petit llenguatge per canviar les variables de la interfície d’interacció persona - computador. Aquest permet activar i desactivar algorismes amb les comandes activate o deactivate seguides del nom de l’algorisme i canviar el valor dels paràmetres mitjançant la comanda set “nom del paràmetre” = valor.
Per implementar el llenguatge es simulen 4 autòmats, el principal i un per cada tipus de comanda.
L’autòmat principal llegeix l’entrada i comprova si el valor que ha llegit des de l'inici coincideix amb alguna de les 3 comandes. En cas que coincideixi es passa a executar l'autòmat de la comanda corresponent.
Tant l'autòmat de la comanda activate com el de la comanda deactivate llegeixen la resta de l’entrada i comproven que el resultat coincideixi amb algun dels algorismes, en cas contrari produeixen una excepció.
L'autòmat de la comanda set llegeix la resta de l’entrada fins que coincideixi amb algun dels paràmetres (A, w, fase0, s). Quan coincideix llegeix el símbol “=” i després llegeix la resta de l’entrada i la converteix en enter si el paràmetre és s i real si el paràmetre és A, w o fase0. En cas que format sigui incorrecte produeix una excepció.

Aprenentatge automàtic

S'ha utilitzar per classificar les comandes de l'intèrpret en cas que l'usuari no les entri correctament.
Per representar el text a classificar s'ha utilitzat un vector de mida fixa amb 26 posicions per les lletres, 10 per als números, un per l'espai, un per la resta de caràcters i 1 per la mida del text. El valor a cada posició és el número de vegades que el símbol apareix en el text. No s'ha distingit entre majúscules i minúscules.
S'han probat el perceptró multicapa, el mètode KNN i els mètode SVM (lineal, polinomic i RDF).
Els 3 mètodes són sensibles a l'escalat de les característiques per això s'han escalat les dades amb StandardScaler de sklearn. S'ha guardat el scaler utilitzat amb el dataset de training per utilitzar el mateix amb les entrades de l'usuari que es vulguin classificar.
Per entrenar els models s'han creat dades artificialment considerant 3 operacions a partir de les dades correctes: insertar un caracter, modificar-lo i esborrar-lo. Hi ha una probabilitat d'aplicar una de les 3 operacions, es genera un número entre 0 i 1 per cada posició de la paraula i si aquest és menor que la probabilitat de corrompre'l llavors s'aplica una de les 3 operacions amb probabilitat uniforme.

MLP

És un mètode supervisat d'aprenentatge automàtic. La diferencia entre l'aprenentatge automatic supervisat i no supervisat es que el primer és que en el supervisat és disposa de les sortides de cada entrada de les dades de training mentre que en el no supervisat no. El perceptró multicapa conté la capa d'entrada, la de sortida i entremig varies capes ocultes no lineals.



Si tenim n mostres de dades, m característiques per cada mostra de dades, k capes ocultes de h neurones cada una, o neurones a la capa de sortida, i l'executem i iteracions, el cost d'entrenar el perceptor multicapa segons la documentació de sklearn és:

$O(n\ m\ h^k o\ i)$

Learning

Cada neurona de les capes ocultes transforma els valors de la capa anterior mitjançant un sumatori lineal amb pesos:

$\nu = x_1 w_1 + x_2 w_2 + ... + x_n w_n$

On x_i son els valors de les capes anteriors i w_i els pesos. El resultat s'aplica a una funció d'activació no lineal:

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1}$$

Consisteix en canviar els pesos w_i després de procesar cada dada depenent de l'error. Sigui e l'error, d el valor real i y el valor produït pel perceptró:

$$e_j(n) = d_j(n) - y_j(n)$$

Els pesos s'ajusten per minimitzar l'error total:

$$\mathcal{E}(n) = \frac{1}{2} \sum_j e_j^2(n)$$

Utilitzant l'algorisme del gradient descendent:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n)$$

On η és el learning rate que s'ha de seleccionar de tal forma que els pesos w_i convergeixin ràpidament cap a una resposta. La derivada depèn de υ_j que es variable. (Recordem que υ_j és el sumatori lineal de pesos i punts anteriors $\upsilon_j = x_1 w_1 + x_2 w_2 + ... + x_n w_n$)

Per la capa de sortida aquesta derivada es pot simplificar per:

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = e_j(n) \phi'(v_j(n))$$

On ϕ ' és la derivada de la funció d'activació.

Per les capes ocultes la derivada és:

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = \phi'(v_j(n)) \sum_k -\frac{\partial \mathcal{E}(n)}{\partial v_k(n)} w_{kj}(n)$$

KNN

És un mètode supervisat d'aprenentatge automàtic no paramètric. En classificació la classe resultant s'obté a partir del vot dels k veïns més propers, assignant-se la classe de la majoria d'aquests veïns.
És un mètode lazy learning. Això significa que només aproxima localment i tota la computació és aplaçada fins a la classificació.
En la fase de training únicament es guarden els exemples en les estructures adequades.
En la fase de predicció es classifica el nou exemple amb la classe més freqüent entre els k veïns més propers. (On k és un hiper paràmetre). Es poden utilitzar pesos donant més influència als veïns més propers.

Complexitat

Sigui N el número de mostres i D el número de característiques de cada mostra.
L'algorisme de força bruta (comprovar la distancia del nou punt amb tots els altres) té complexitat $O(DN^2)$.
Per millorar l'eficiència s'utilitzen estructures de dades d'arbres que redueixen el número de càlculs de distàncies necessaris. Es basen en que si un punt A és molt lluny d'un punt B i un punt C és proper al punt B, el punt C i el punt A són molt llunyans. La complexitat és $O(DNlog(N))$.

K-D Tree

Generalitza els Quad-trees a qualsevol número de dimensions. És un arbre binari que divideix recursivament l'espai en dues regions ortotropiques anidades on s'insereixen els punts. La construcció és ràpida perquè només es divideix l'espai a partir del eixos cartesians. Un cop construït el veí més proper es pot trobar amb $O(log(N))$ càlculs de distàncies. Aquesta estructura de dades esdevé ineficient per a dimensions grans (D > 20).

Ball Tree

A diferencia dels K-D Trees aquests divideixen l'espai a través de hiper-esferes anidades. Cada esfera es defineix per un centroide C i un radi r totes els punts en un node es troben a dintre la hiper-esfera. El número de punts candidats es redueix degut a $|x+y| \leq |x| + |y|$. Utilitzant això si calculem la distància entre un punt p i el centroide obtenim una cota inferior i superior de la distancia entre el punt p i tots els punts del node.

SVM

És un mètode d'aprenentatge automàtic supervisat no probabilístic.
Un model és una representació dels punts en l'espai separats en 2 o més classes per un hiperplà o conjunt de hiperplans. El millor hiperplà és aquell que té la distància més gran al punt més proper de qualsevol classe perquè en general aquest hiperplà té menor error de generalització.
Com que a vegades el problema pot no ser linealment separable l'espai original es pot mapejar en un altre de major dimensió. Amb això es pot aconseguir fer el problema més senzill en aquesta dimensió.
Tenim un conjunt de punts:

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n),$$

On $\widehat{x_i}$ és un vector de p dimensions i y_i pot valdre 1 o -1 i indica a quina classe pertany $\widehat{x_i}$. Volem trobar el hiperplà de màxim marge que separa els punts amb y = 1 dels punts amb y = -1, es el hiperplà tal que la distancia entre el hiperplà i el punt més proper a aquest de qualsevol classe és màxima.
Podem escriure un hiperplà com:

$$\vec{w} \cdot \vec{x} - b = 0,$$

On $\overline{w}$ és el vector normal al hiperplà.

Hard margin

Si les dades són linealment separables, agafem 2 hiperplans paral·lels que separen les dos classes tal que la distancia entre ells és màxima. El hiperplà de màxim marge es el que està al mig. Els 2 hiperplans es poden descriure:

$$\vec{w} \cdot \vec{x} - b = 1$$

$$\vec{w} \cdot \vec{x} - b = -1$$

La distància entre aquests 2 hiperplans és $\frac{2}{||\widehat{w}||}$, així per maximitzar la distancia necessitem minimitzar $||\widehat{w}||$. Per tots els punts tenim les següents equacions:
$\widehat{w}.\widehat{x} - b \geq 1 \; si \; y_i = 1$
$\widehat{w}.\widehat{x} - b \leq -1 \; si \; y_i = -1$
Es pot reescriure com:
$y_i(\widehat{w}.\widehat{x_i} - b) \geq 1 \, for \; 1 \leq i \leq n$
Juntant-ho obtenim el següent problema d'optimització:
minimize $||\widehat{w}||$ subject to $y_i(\widehat{w}.\widehat{x_i} - b) \geq 1 \, for \; 1 \leq i \leq n$
El hiperplà de màxim marge queda determinat només per els punts més propers a aquest. Aquests punts s'anomenen support vectors.

Soft margin

Per als casos on les dades no són linealment separables introduïm la hinge loss function:

$$\max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right).$$

Aquesta funció és 0 si la dada en qüestió cau al cantó correcte del marge. Si cau a la banda incorrecte la funció és proporcional a la distancia del marge. Volem minimitzar:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right)\right] + \lambda\|\vec{w}\|^2,$$

On λ determina el trade-off entre augmentar la mida del marge i assegurar-se que x_i cau a la banda correcte del marge.

Algorisme

Primal

Introduïm la variable:

$$\zeta_i = \max\left(0, 1 - y_i(w \cdot x_i - b)\right)$$

Fent aquest canvi el problema d'optimització queda:

$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n}\zeta_i + \lambda\|w\|^2$$

$$\text{subject to } y_i(w \cdot x_i - b) \geq 1 - \zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i.$$

Dual

Solucionant el Lagrangià dual, obtenim el següent problema:

$$\text{maximize } f(c_1 \ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j,$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.$$

Aquest és una funció quadràtica de les c_i subjecte a restriccions lineals es pot resoldre eficientment mitjançant algorismes de programació quadràtica.
Les variables c_i es defineixen de forma que:

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \vec{x}_i$$

$c_i = 0$ quan el punt x_i es troba a la banda correcte del marge. Quan x_i es troba al marge $0 < c_i < (2n\lambda)^{-1}$.

El offset b es pot recuperar una x_i al límit del marge i solucionant la següent equació.

$$y_i(\vec{w} \cdot \vec{x}_i - b) = 1 \iff b = \vec{w} \cdot \vec{x}_i - y_i.$$

Kernel

$$k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)$$

El vector $\widehat{w}$ en l'espai transformat satisfà:

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \varphi(\vec{x}_i),$$

On c_i es troben resolent el problema d'optimització:

$$\begin{aligned} \text{maximize } f(c_1 \dots c_n) &= \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (\varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)) y_j c_j \\ &= \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i k(\vec{x}_i, \vec{x}_j) y_j c_j \end{aligned}$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.$$

Podem trobar un índex i tal que $0 < c_i < (2n\lambda)^{-1}$ així $\varphi(x_i)$ està al límit del marge de l'espai transformat. Després podem trobar b

$$\begin{aligned} b = \vec{w} \cdot \varphi(\vec{x}_i) - y_i &= \left[\sum_{j=1}^{n} c_j y_j \varphi(\vec{x}_j) \cdot \varphi(\vec{x}_i) \right] - y_i \\ &= \left[\sum_{j=1}^{n} c_j y_j k(\vec{x}_j, \vec{x}_i) \right] - y_i. \end{aligned}$$

$$\vec{z} \mapsto \operatorname{sgn}(\vec{w} \cdot \varphi(\vec{z}) - b) = \operatorname{sgn}\left(\left[\sum_{i=1}^{n} c_i y_i k(\vec{x}_i, \vec{z}) \right] - b \right).$$

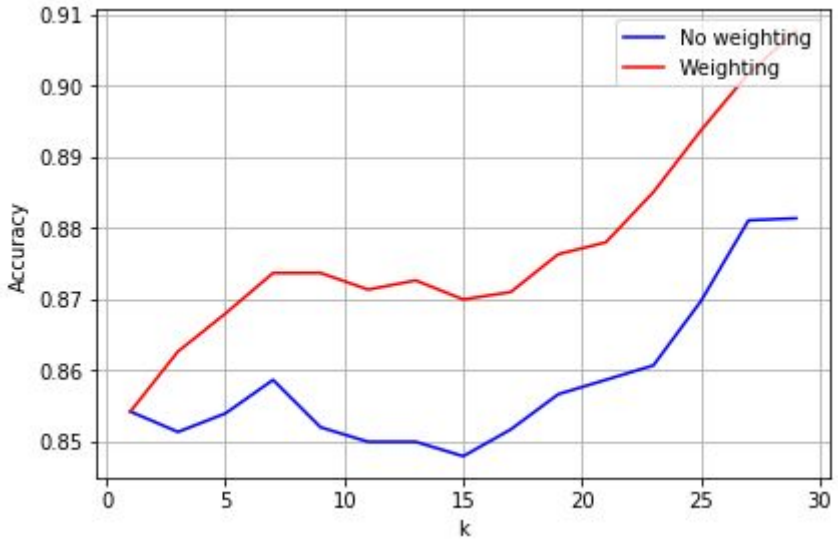
Complexitat

La part més importat és resoldre el problema de programació quadràtica. La implementació de sklearn té una complexitat entre $O(n_{features}n^2_{samples})$ i $O(n_{features}n^3_{samples})$ depenent de les dades.

Paràmetres

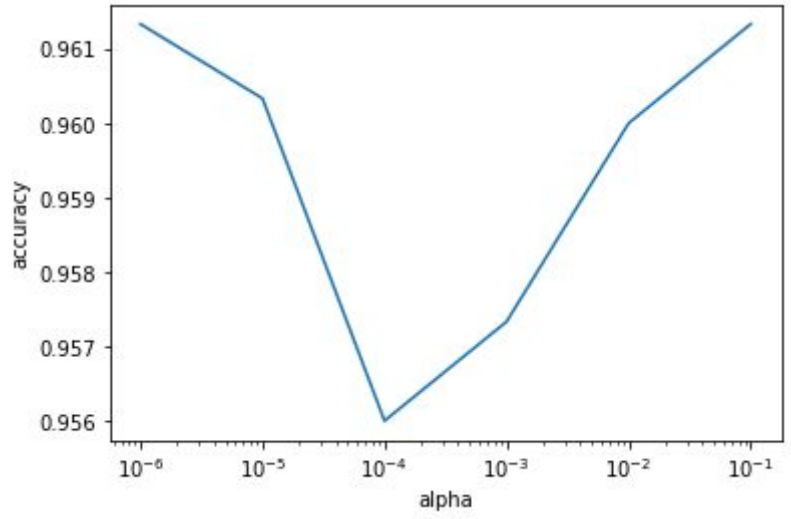
KNN

Els millors paràmetres són 30 veïns i fer el càlcul amb pesos.



MLP

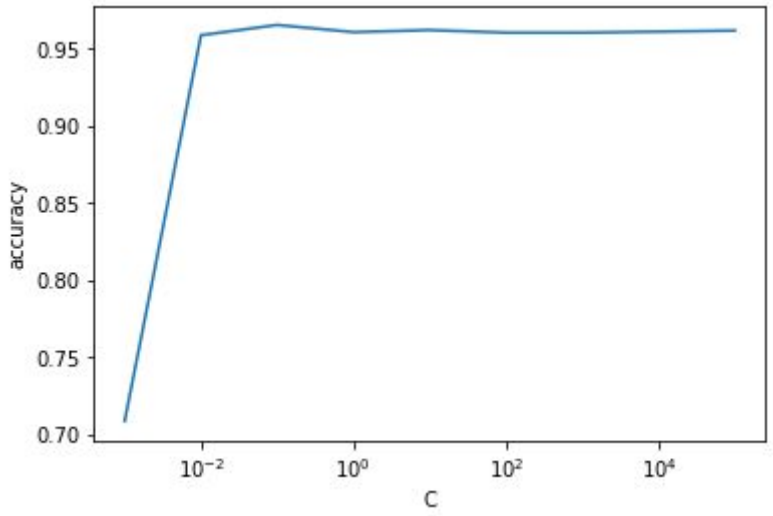
Els millors paràmetres són alpha 0.1 amb una accuracy de 0.961



SVM

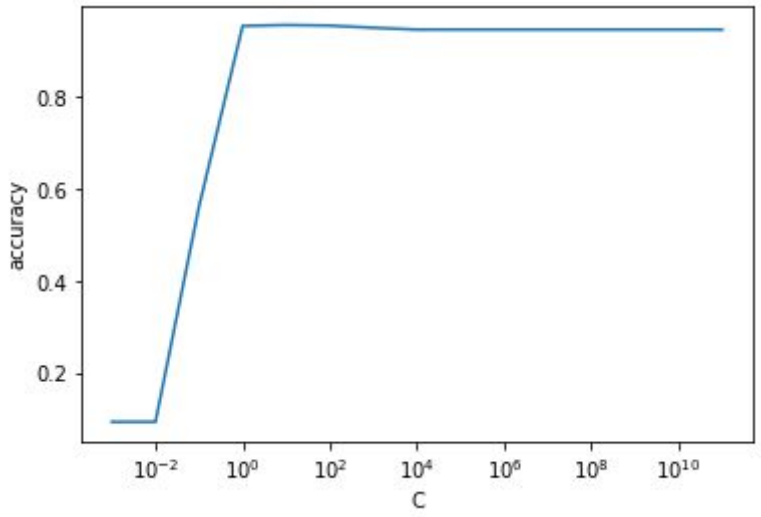
Linear

Els millors paràmetres són C = 0.1 amb una accuracy de 0.965.



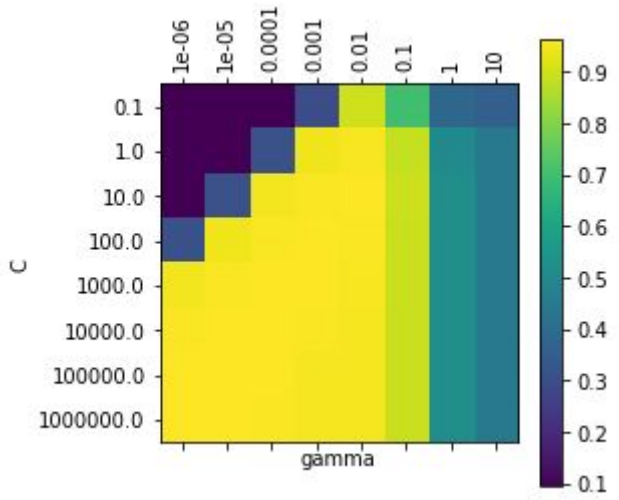
Polynomial

Els millors paràmetres són C = 10 amb una accuracy de 0.955



RDF

Els millors paràmetres són C = 100, gamma = 0.001 amb una accuracy de 0.964



Comparació

Observació	Els diferents algorismes d'aprenentatge automàtic poden tenir diferent precisió i temps d'aprenentatge i predicció.
Plantejament	Mesurarem els temps d'aprenentatge i predicció i la precisió dels 5 algorismes d'aprenentatge automàtic.
Hipòtesis	Els 5 algorismes són similars (H0) o són diferents i n'hi ha un millor
Mètode	● Generarem 10 conjunts de dades de training i test aleatoris. ● Mesurarem el temps de training i test i la precisió per als 5 algorismes per cada problema.

	KNN	MLP	SVM linear	SVM poly	SVM RDF
esperance training time	0.038	7.29	0.1969	0.3896	0.214
deviation training time	0.011	0.24096	0.0345	0.02867	0.004298
esperance predict time	1.77	0.01168	0.261	0.39768	0.342
deviation predict time	0.03	0.0028	0.00458	0.00899	0.00491
esperance accuracy	0.8979	0.9644	0.9663	0.9588	0.9663
deviation accuracy	0.007157	0.0025	0.00151	0.00297	0.001815

SVM lineal es millor que SVM poly i que SVM RDF ja que té un menor temps de training i test i una precisió similar. KNN té una menor precisió que els 4 altres algorismes. MLP té un temps de training molt alt cosa que provoca que l'aplicació trigui més a iniciar-se. Per tots aquests motius utilitzarem SVM lineal.

Informe de sostenibilitat

Coneixo a grans trets les causes i conseqüències sobre la problemàtica social economica i ambiental però no les solucions. Se analitzar la sostenibilitat desde la perspectiva d'aquestes dimensions però no sé com medir-la ni coneixo les tecnologies "sostenibilistes" aplicables a un projecte TIC. Se relacionar un problema amb altres ja coneguts, és a dir reduir un problema a un altre. Tinc creativitat però no utilitzo cap estrategia per generar idees, a vegades se m'acudeixen idees per millorar la sostenibilitat. Se que les TIC poden millorar la sostenibilitat en tots els seus aspectes. Coneixo els problemes de justicia social, equitat, diversitat i transparència però no entenc la relació dels tres primers amb les TIC, per això no els aplico. Se valorar com afecta a la societat un producte de les TIC, si fa la vida més fàcil a la població. Crec que si un producte fa la vida més fàcil a la societat llavors millora el bé comú. Entenc la necessitat de l'accessibilitat, l'ergonomia i la seguretat en els productes TIC. Quan treballo en un projecte em limito a complir amb la feina que em demanen i no em preocupo pel bé comú. Comprenc les parts del pressupost d'un projecte com les amortitzacions, els costos fixos i variables … No conec els principis deontologics, però tinc entès que estan relacionats amb la moral.

Dimensió econòmica

Respecte el Projecte posat en producció (PPP): Reflexió sobre el cost que has estimat per la realització del projecte?

- Com he estimat en l'apartat del pressupost el cost del projecte es d'uns 5000 euros. Aproximadament uns 3600 en recursos humans i la resta en recursos materials.

Respecte la Vida Útil: Cóm es resolen actualment els aspectes de costos del problema que vols abordar (estat de l'art)?

- Actualment el problema es resol amb l'algorisme de Verlet. Els ordinadors són molt potents i aquest algorisme és poc costos, per tant, actualment no hi ha problema.

En què millorarà econòmicament (costos...) la teva solució respecte a les existents?

- El meu treball consisteix en modificar-lo cosa que difícilment millorarà la situació econòmica, a no ser que la modificació resulti molt eficient o millori molt la presició.

Dimensió ambiental

Respecte el Projecte posat en producció (PPP): ¿Has estimat el impacte ambiental que tindrà la realització del projecte?

- L'únic impacte ambiental del meu projecte és el consum d'electricitat de l'ordinador, què és un impacte molt petit.

Respecte el Projecte posat en producció (PPP):¿T'has plantejat minimitzar-ne el impacte, per exemple, reutilitzant recursos?

- Per minimitzar l'impacte s'utilitzarà python què és un llenguatge de scripting que permet una implementació més ràpida. Cosa que es tradueix a menys hores d'ús de l'ordinador i per tant menys consum elèctric.

Respecte la Vida Útil: Cóm es resol actualment el problema que vols abordar (estat de l'art)?, i. ¿En què millorarà ambientalment la teva solució respecte a les existents?

- Igual que en la dimensió econòmica, si la modificació no millora molt en eficiencia no es produirà cap millora ambiental.

Dimensió Social

Respecte el Projecte posat en producció (PPP): Què creus que t'aportarà a nivell personal la realització d'aquest projecte?

- Aprendre sobre mètodes numèrics per resoldre equacions diferencials i sobre física.

Respecte la Vida Útil: Cóm es resol actualment el problema que vols abordar (estat de l'art)? i. ¿En què millorarà socialment (qualitat de vida) la teva solució respecte les existents?

- No millorarà la qualitat de vida.

Respecte la Vida Útil: Existeix una necessitat real del projecte?

- No. Els algorismes que s'apliquen ja són prou eficients tenint en compte la potència de càlcul dels ordinadors actuals.

Referències

M.P.Allen, D.J.Tildesley. Computer Simulation of Liquids. Oxford Science Publications. 1987. ISBN 0198556454, 9780198556459.
Daan Frenkel, Berend Smith, Understanding Molecular Simulation. Academic Press. 2002. ISBN 978-0-12-267351-1
Anàlisi numèric. A wikipedia [en línia], Wikimedia Foundation 2019. [Consulta: 16 setembre 2019] Disponible a: https://es.wikipedia.org/wiki/An%C3%A1lisis_num%C3%A9rico
Equacions diferencials. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta: 16 setembre 2019] Disponible a: https://es.wikipedia.org/wiki/An%C3%A1lisis_num%C3%A9rico#Ecuaciones_diferenciales
Mètode d'Euler. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 16 setembre 2019] Disponible a: https://en.wikipedia.org/wiki/Euler_method
https://es.wikipedia.org/wiki/M%C3%A9todo_de_Euler
Serie de Taylor. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 16 setembre 2019] Disponible a: https://es.wikipedia.org/wiki/Serie_de_Taylor
Algorisme de Verlet. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 16 setembre 2019] Disponible a: https://en.wikipedia.org/wiki/Verlet_integration
Algorisme de Verlet leap-frog. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 16 setembre 2019] Disponible a: https://en.wikipedia.org/wiki/Leapfrog_integration
Mètodes de Runge Kutta. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 16 setembre 2019] Disponible a: https://es.wikipedia.org/wiki/M%C3%A9todo_de_Runge-Kutta
Mètodes de múltiples pasos. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 12 novembre 2019] Disponible a: https://en.wikipedia.org/wiki/Linear_multistep_method#Families_of_multistep_methods
Funció de distribució radial. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 5 desembre 2019] Disponible a: https://en.wikipedia.org/wiki/Radial_distribution_function
https://es.wikipedia.org/wiki/Funci%C3%B3n_de_distribuci%C3%B3n_radial
Difusió molecular. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 6 desembre 2019] Disponible a:

https://en.wikipedia.org/wiki/Molecular_diffusion
Funció d'autocorrelació de velocitats. A nanohub.org [en línia], A. Martini. [Consulta 5 desembre 2019] Disponible a: https://nanohub.org/resources/7581/download/Martini_L9_DynamicProperties.pdf
Desplaçament quadràtic mitjà. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 6 desembre 2019] Disponible a: https://en.wikipedia.org/wiki/Mean_squared_displacement
Mètode predictor-corrector. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 7 desembre 2019] Disponible a: https://en.wikipedia.org/wiki/Predictor%E2%80%93corrector_method
Tutorial de opengl. A learnOpenGL [en línia], Joey de Vries. [Consulta 14 desembre 2019] Disponible a: https://learnopengl.com/
MLP. A scikit-learn [en línia]. [Consulta 31 desembre 2019] Disponible a: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#classification
MLP. A wikipedia [en línia], Wikimedia Foundation 2019 [Consulta 31 desembre 2019] Disponible a: https://en.wikipedia.org/wiki/Multilayer_perceptron
KNN. A wikipedia [en línia], Wikimedia Foundation 2019 [Consula 5 gener 2020] Disponible a: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
KNN. A scikit-learn [en línia]. [Consulta 5 gener 2020] Disponible a: https://scikit-learn.org/stable/modules/neighbors.html
SVM. A wikipedia [en línia], Wikimedia Foundation 2019 [Consula 7 gener 2020] Disponible a: https://en.wikipedia.org/wiki/Support-vector_machine
SVM. A scikit-learn [en línia]. [Consulta 9 gener 2020]. Disponible a: https://scikit-learn.org/stable/modules/svm.html
Jornada laboral anual. A unCOMO [en línia], linktoMEDIA [Consulta 25 setembre 2019] Disponible a: https://negocios.uncomo.com/articulo/como-calcular-la-jornada-laboral-anual-26339.html
Salari d'un gestor de projectes. A indeed [en línia], indeed [Consulta 25 setembre 2019] Disponible a: https://www.indeed.es/salaries/Gestor/a-de-proyectos-Salaries
Salari d'un enginyer informàtic. A universidad europea [en línia], universidad europea [Consulta 15 gener 2019] https://universidadeuropea.es/blog/cuanto-gana-un-ingeniero-informatico
Salari d'un cap de projectes. A indeed [en línia], indeed [Consulta 25 setembre 2019] Disponible a: https://www.indeed.es/salaries/Jefe-de-proyecto-Salaries

Preu d’un ordinador. A mediamarkt [en línia], MediaMarkt [Consulta 25 setembre 2019] Disponible a: https://www.mediamarkt.es/es/category/_ordenadores-701427.html
Consum d’un ordinador. A hardzone [en línia],Hard Zone [Consulta 25 setembre 2019] Disponible a: https://hardzone.es/2015/03/31/cuanto-cuesta-la-electricidad-que-consume-tu-pc/
Preu kw/h. A lucera [en línia], lucera [Consulta 25 setembre 2019] Disponible a: https://lucera.es/tarifas-luz?adw_campaign=1597293729&adw_adgroupid=61500662558&adw_keyword=&gclid=Cj0KCQjwoKzsBRC5ARIsAITcwXEYU7RzAemSUWoMGx9rT0tIL1xyDqZ8_iae8WQjHo1bmvqIYP2RyxgaAo9EEALw_wcB